A New Dynamic Programming Approach for Spanning Trees with Chain Constraints and Beyond

Martin Nägele[*]   Rico Zenklusen[†]

Abstract

Short spanning trees subject to additional constraints are important building blocks in various approximation algorithms, and, moreover, they capture interesting problem settings on their own. Especially in the context of the Traveling Salesman Problem (TSP), new techniques for finding spanning trees with well-defined properties have been crucial in recent progress. We consider the problem of finding a spanning tree subject to constraints on the edges in a family of cuts forming a laminar family of small width. Our main contribution is a new dynamic programming approach where the value of a table entry does not only depend on the values of previous table entries, as it is usually the case, but also on a specific representative solution saved together with each table entry. This allows for handling a broad range of constraint types.

In combination with other techniques—including negatively correlated rounding and a polyhedral approach that, in the problems we consider, allows for avoiding potential losses in the objective through the randomized rounding—we obtain several new results. We first present a quasi-polynomial time algorithm for the Minimum Chain-Constrained Spanning Tree Problem with an essentially optimal guarantee. More precisely, each chain constraint is violated by a factor of at most $1 + \varepsilon$, and the cost is no larger than that of an optimal solution not violating any chain constraint. The best previous procedure is a bicriteria approximation violating each chain constraint by up to a constant factor and losing another factor in the objective. Moreover, our approach can naturally handle lower bounds on the chain constraints, and it can be extended to constraints on cuts forming a laminar family of constant width.

Furthermore, we show how our approach can also handle parity constraints as used in the context of (path) TSP and a generalization thereof, and discuss implications in this context.

---

[*]Department of Mathematics, ETH Zurich, Zurich, Switzerland. Email: martin.naegele@ifor.math.ethz.ch.

[†]Department of Mathematics, ETH Zurich, Zurich, Switzerland. Email: ricoz@math.ethz.ch. Supported by Swiss National Science Foundation grant 200021_165866.

1 Introduction

Given a graph $G = (V, E)$ and edge costs $c: E \to \mathbb{R}_{\geqslant 0}$, the problem of finding a minimum cost spanning tree (MST) in G with respect to c is one of the most classical network design problems. A variety of applications in areas like chip design, vehicle routing, and telecommunication networks triggered interest in constrained spanning tree problems. Moreover, such problems are regularly used as building blocks in the design of approximation algorithms. In particular, many approaches used in recent progress on the Traveling Salesman Problem (TSP) and its path version have a constrained spanning tree problem as a key component.

The arguably most classical example of a constrained spanning tree problem is the minimum bounded degree spanning tree problem (MBDST). Here, the goal is to find a spanning tree $T \subseteq E$ in G of minimum cost subject to T satisfying a degree constraint $|T \cap \delta(v)| \leqslant d(v)$ at every vertex v, where $d: V \to \mathbb{Z}_{>0}$ are given degree bounds. Already just finding a feasible solution for MBDST can easily be seen to be NP-hard, even in the special case where $d(v) = 2$ for all v, as this captures the Hamiltonian path problem. This is typical for most constrained spanning tree problems. The focus has therefore been on approximation algorithms that allow for a slight violation of the additional constraints. This led to algorithms with various trade-offs between cost and constraint violation. After a series of papers with progress on the approximation guarantees (see [17, 18, 14, 5, 6] and references therein), an essentially best possible approximation algorithm for MBDST was given by Singh and Lau [27]. Using iterative relaxation, they return a spanning tree violating each degree constraint by at most 1 unit, and of cost no more than that of an optimal solution not violating the degree constraints. Bansal, Khandekar, and Nagarajan [4] presented an elegant generalization of this result to upper bounds on the number of edges picked in a family of arbitrary edge sets $E_1, \ldots, E_k \subseteq E$. More precisely, they show that a spanning tree violating each constraint by at most $\max_{e \in E} |\{i \in [k] \mid e \in E_i\}| - 1$ and with cost no more than that of an optimal solution can be found. If each edge is only contained in a constant number of constraints, this still leads to a constraint violation by only an additive constant. Whereas iterative relaxation is undoubtedly a very strong tool to find constrained spanning trees, it is difficult to obtain constraint violations of at most a constant (either additively or multiplicatively) through this technique when edges can be in a super-constant number of constraints (see [33] for one rare example of this type).

However, constrained spanning tree problems appearing in the design of approximation algorithms, especially within problems related to TSP, are often of this type. (When referring to TSP and its variants, we always assume that the involved lengths are metric.) For example, Asadpour, Goemans, Madry, Oveis Gharan, and Saberi [2] established a beautiful connection between the asymmetric TSP and so-called thin trees, which are trees with constraints on all cut sets. More precisely, if there is a constant c such that for any $k \in \mathbb{Z}_{>0}$, one can efficiently find in any k-edge-connected graph $G = (V, E)$ a spanning tree $T \subseteq E$ with $|T \cap \delta(S)| \leqslant c/k \cdot |\delta(S)|$ for all $S \subseteq V$, then this can be transformed into an $\mathcal{O}(1)$-approximation for asymmetric TSP. Such trees are sometimes referred to as constantly-thin trees. The existence of a weaker version of constantly-thin trees was conjectured by Goddyn [13] and remains open. We highlight that very recently, Svensson, Tarnawski, and Végh [28] obtained a 5500-approximation for asymmetric TSP through different techniques. They state that even though a more careful analysis may bring the approximation factor down to the order of hundreds, different ideas are needed to obtain considerably smaller factors. Finding constantly-thin spanning trees may be one path to advance on this question. Moreover, especially for Path TSP, where the task is to find a shortest Hamiltonian s-t path in a complete graph with metric lengths, finding spanning trees with various additional constraints/properties has been crucial in recent progress [1, 25, 31, 15, 26, 30, 34]. Interestingly, the type of tree properties considered for Path TSP are often on the edges contained in a family of s-t cuts that form a chain.[1] More generally, the shortest connected T-join problem, which generalizes both Path TSP and classical TSP, naturally leads to a laminar family of cuts to be considered [9].[2] The appearance of cut families with laminar or chain structure in this context

---

[1] Throughout this paper, a *cut* of a vertex set V is a nonempty set $S \subsetneq V$. An edge e *lies in* a cut S if $e \in \delta(S)$.

[2] For some even cardinality vertex set $T \subseteq V$ in a graph $G = (V, E)$, a T-join is an edge set $U \subseteq E$ such that the vertices of odd

1

stems from the use of combinatorial uncrossing arguments, which are ubiquitous in the context of TSP, and is thus not surprising. Clearly, when constraints are imposed on the edges in a family of cuts that are laminar, or even just a chain, then edges can appear in a large number of constraints.

The arguably most canonical constrained spanning tree problem with constraints on a laminar family of cuts is when there are upper bounds on the number of edges in each cut. This setting was considered by Bansal, Khandekar, Könemann, Nagarajan, and Peis [3], who designed an iterative relaxation approach for returning a spanning tree violating each constraint by at most $\mathcal{O}(\log |V|)$ units and being of cost no more than the cost of an optimal solution not violating the constraints. As later shown by Olver and Zenklusen [23], this is almost optimal because an additive violation of $c \log |V| / \log \log |V|$ units, for some constant c, cannot be achieved unless $\mathrm{P} = \mathrm{NP}$. It remains open whether $\mathcal{O}(1)$-multiplicative violations are possible.

In summary, constrained spanning tree problems where edges can appear in a large number of constraints are still badly understood, and new approaches and techniques are needed.

The goal of this paper is to introduce a versatile dynamic programming type approach to deal with a variety of constraint types on laminar cut families of small width, with applications to chain-constrained spanning trees, Path TSP and beyond. Dynamic programming did not play a crucial role in the above-mentioned problems until a very recent breakthrough result by Traub and Vygen [30] in the context of Path TSP, which inspired this work. A key new technical ingredient in our approach is to introduce a generalized form of dynamic programming, where the value of a table entry does not only depend on the values of previous table entries, as it is usually the case, but also on a fixed representative solution saved together with each table entry. This leads to the peculiar situation that it is hard to define upfront the solution set over which our dynamic program optimizes. However, we can show that it optimizes over a relaxation of the problems we are interested in, and returns solutions with well-defined properties to be exploited later on, which is all we need. For chain-constrained problems, our dynamic program can be leveraged to return a fractional point in the spanning tree polytope, which can then be rounded to an actual spanning tree. We show that good spanning trees can be obtained by using negatively correlated rounding procedures together with an alteration procedure that may be of independent interest, and which we therefore present in a more general context.

1.1 Our results

Here, we provide a summary of the results that we obtain by combining our dynamic programming approach with various other techniques. We start with a natural special case of laminarly constrained spanning trees that has been studied previously, namely the *minimum chain-constrained spanning tree* problem (MCCST), where upper bounds are imposed on the number of edges that can be chosen in a family of cuts that form a chain. Opposed to previous results, we also allow for lower bounds on the number of edges in the cuts, which can be handled with our methods without additional complications.

Minimum Chain-Constrained Spanning Tree Problem (MCCST): Let $G = (V, E)$ be a graph with edge costs $c \colon E \to \mathbb{R}_{\geqslant 0}$, and let $\emptyset \subsetneq S_1 \subsetneq S_2 \subsetneq \ldots \subsetneq S_k \subsetneq V$ and $a_1, \ldots, a_k, b_1, \ldots, b_k \in \mathbb{Z}_{>0}$. Find a spanning tree $T \subseteq E$ minimizing $c(T) := \sum_{e \in T} c(e)$ among all trees satisfying

$$a_i \leqslant |T \cap \delta(S_i)| \leqslant b_i \quad \text{for all } i \in [k] := \{1, \ldots, k\}.$$

For $\alpha, \beta \geqslant 1$, we say that an algorithm returning a spanning tree T is an (α, β)-*approximation* for MCCST, if $\frac{1}{\beta} \cdot a_i \leqslant |T \cap \delta(S_i)| \leqslant \beta \cdot b_i$ for all $i \in [k]$, and $c(T) \leqslant \alpha \cdot c(\mathrm{OPT})$, where OPT is a spanning tree of minimum cost among all spanning trees not violating the chain constraints. For MCCST without lower bounds, i.e., $a_1 = \ldots = a_k = 0$, Linhares and Swamy [20] recently presented an efficient $(\frac{\lambda}{\lambda-1}, 9\lambda)$-approximation

---

degree in the subgraph (V, U) are precisely T. Moreover, in the shortest connected T-join problem, one is allowed to choose as U a multiset of edges in E.

for any $\lambda > 1$ by extending a prior approach of Olver and Zenklusen [23] that did not handle costs. Our main result is a quasi-polynomial algorithm for MCCST (with lower bounds) with essentially best possible guarantees.

Theorem 1. *For every $\varepsilon > 0$, there is a randomized $(1, 1 + \varepsilon)$-approximation algorithm for MCCST with running time $|V|^{\mathcal{O}(\log|V|/\varepsilon^2)}$.*

The approximation guarantee is essentially best possible in the sense that finding a tree that fulfills all chain constraints is NP-hard as shown in [23]. Our randomized algorithm returns a $(1, 1 + \varepsilon)$-approximation with high probability, and can also be transformed into a Las Vegas algorithm. Moreover, Theorem 1 gives hopes that such best possible guarantees may be achievable with an efficient procedure. We prove the theorem through a combination of our new dynamic programming approach, which we will introduce in this context, together with negatively correlated rounding and an alteration step to improve the value of the final solution. As we discuss in Section 4, in this context, the alteration step we use could also be replaced by a technique introduced by Linhares and Swamy [20].

It turns out that with some modifications, the technical insights outlined above are enough to push our results beyond pure chain constraints towards the more general problem of *minimum laminarly constrained spanning trees* (MLCST), which is defined as follows.

> **Minimum Laminarly Constrained Spanning Tree Problem (MLCST):** Let $G = (V, E)$ be a graph with edge costs $c \colon E \to \mathbb{R}_{\geqslant 0}$, let $\mathcal{L} \subseteq 2^V \setminus \{\emptyset, V\}$ be a laminar family, and $a_S, b_S \in \mathbb{Z}_{>0}$ for $S \in \mathcal{L}$. Find a spanning tree $T \subseteq E$ minimizing $c(T) := \sum_{e \in T} c(e)$ among all trees satisfying
> $$a_S \leqslant |T \cap \delta(S)| \leqslant b_S \quad \text{for all } S \in \mathcal{L}.$$

Note that if $\mathcal{L}$ contains precisely all singletons, then the above problem setting reduces to the minimum bounded degree spanning tree problem (MBDST) mentioned in the introduction. From a structural point of view, constraint types in the special cases MCCST and MBDST are "orthogonal" in the sense that the role of $\mathcal{L}$ is taken by a chain in one case and by an antichain in the other. Currently, there is no efficient approach covering both cases. For a step towards MLCST using our dynamic programming framework, we parametrize laminar families by their *width*, which is the smallest integer k such that the laminar family does not contain any $k + 1$ disjoint sets. Using this notion, we can generalize Theorem 1 to obtain the following result.

Theorem 2. *For every $\varepsilon > 0$, there is a randomized $(1, 1 + \varepsilon)$-approximation algorithm for MLCST with running time $|V|^{\mathcal{O}(k \log|V|/\varepsilon^2)}$, where k is the width of the laminar family $\mathcal{L}$.*

Observe that the running time in the above result depends exponentially on the width of the laminar family. For width k up to the order $\mathcal{O}(\log|V|)$, we thus still achieve quasi-polynomial running time. Unfortunately, the exponential dependence on k seems to be intrinsic to our approach, and most likely, new ideas are required to overcome this.

Our dynamic programming approach is very versatile in terms of constraint types that can be handled. To highlight this fact, we show how it can be employed to replicate a recent result of Traub and Vygen [30] in the context of Path TSP.

> **Shortest Hamiltonian s-t Path Problem (Path TSP):** Let $G = (V, E)$ be a complete graph with metric edge lengths $\ell \colon E \to \mathbb{R}_{\geqslant 0}$, and let $s, t \in V$ be two distinct vertices. Find a path $P \subseteq E$ minimizing $\ell(P) := \sum_{e \in P} \ell(e)$ among all Hamiltonian s-t paths in G.

Leveraging our dynamic programming ideas, we obtain the following theorem.

Theorem 3. *For every $\varepsilon > 0$, there is a $(1.5 + \varepsilon)$-approximation algorithm for Path TSP with running time* $|V|^{\mathcal{O}(1/\varepsilon)}$.

The above result matches the algorithm by Traub and Vygen [30] in terms of approximation ratios, but provides a weaker running time guarantee, as Traub and Vygen are able to achieve an exponential dependence on $\log(1/\varepsilon)$. Our approach, on the other hand, avoids recursive calls to a dynamic program as used in [30]. Both results almost match the currently best known approximation guarantee of 1.5, which was recently achieved by Zenklusen [34].

To obtain Theorem 3, our dynamic programming approach can be used to obtain a good spanning tree T for a Christofides-type algorithm, which starts with a spanning tree T and does parity correction in a second step by adding further edges. As observed by An, Kleinberg, and Shmoys [1], it turns out that when following Wolsey's analysis [32] of Christofides algorithm, then the number of edges of T in a chain of cuts crucially impacts the approximation guarantee. This naturally leads to a constrained spanning tree problem for finding T. As for MCCST, the constraints appearing in this problem are on a chain of cuts. However, they are not simply upper and lower bounds—ideally, we would like to impose parity constraints, requiring that each cut is crossed by an odd number of edges. Unfortunately, this leads to NP-hard problems, and we thus use a proxy for parity constraints that is good enough for finding short s-t paths. Again, we hope that this way of dealing with parity constraints, which naturally appear in the context of TSP, may find broader applications.

While techniques from MCCST have implications in Path TSP, their generalization to MLCST can be used in the following generalization of Path TSP.

Metric Shortest Connected T-Join Problem: Let $G = (V, E)$ be a complete graph with metric edge lengths $\ell \colon E \to \mathbb{R}_{\geqslant 0}$, and let $T \subseteq V$ be nonempty with $|T|$ even. Find a T-join $J \subseteq E$ minimizing $\ell(J) \coloneqq \sum_{e \in J} \ell(e)$ among all T-joins J such that (V, J) is connected.

Contrary to Path TSP, here we obtain a constrained spanning tree problem with constraints not only on a chain of cuts, but on a laminar family of width at most $|T| - 1$. These can be handled similarly as in MLCST, giving the following result.

Theorem 4. *For every $\varepsilon > 0$, there is a $(1.5 + \varepsilon)$-approximation algorithm for the metric shortest connected T-join problem with running time* $|V|^{\mathcal{O}(|T|/\varepsilon)}$.

We remark that both the algorithms by Traub and Vygen [30] and Zenklusen [34] can be generalized to the shortest connected T-join problem in a similar way. Hence, for constant $|T| > 4$, all three approaches imply an efficient method improving on the previously best 1.6-approximation by Sebő [25], with the generalization of the approach of Zenklusen giving the currently best known guarantee of 1.5. Nevertheless, we expand on our approach here in order to highlight another immediate implication of our new techniques.

Organization of the paper We start by introducing our techniques in the context of MCCST. Section 2 clearly outlines what we want to achieve with our dynamic program, and why this implies Theorem 1 together with negatively correlated rounding procedures and the solution alteration technique mentioned ealier. Section 3 then provides a thorough discussion of the key aspects of our dynamic programming technique. Section 4 contains additional details on the local alteration approach that we use to obtain a unicriteria approximation for MCCST, and shows a further application of this technique to turn bicriteria approximations into unicriteria ones. In Section 5, we discuss in detail why the natural generalization of our techniques to laminar constraint families fails, and how these difficulties can be overcome to obtain results for MLCST and a proof of Theorem 2. Section 6 presents some implications of our new technique in Path TSP and its generalization, the connected T-join problem, leading to Theorems 3 and 4. Appendix A discusses why, for MCCST, the natural LP relaxation is not strong enough to obtain results with guarantees

as in Theorem 1, thus further motivating the use of a dynamic programming approach to strengthen the relaxation. Finally, Appendix B presents an example showing that a classical analysis of our DP, namely by backtracing an optimal solution, is impossible in the laminarly constrained setting.

2   Overview of our approach for MCCST

The first step of our approach for MCCST relies on finding a solution to a suitable polyhedral relaxation. The canonical relaxation, which was also used in prior results on chain-constrained trees [20, 23], enhances the spanning tree polytope P_{ST} with the cut constraints. We recall that P_{ST} is the convex hull of all characteristic vectors of spanning trees in $G = (V, E)$, and, by a seminal result of Edmonds [11], can be described by

$$P_{\mathrm{ST}} := \left\{ x \in \mathbb{R}_{\geqslant 0}^E \,\middle|\, \begin{array}{c} x(E) = |V| - 1 \\ x(E[S]) \leqslant |S| - 1 \quad \forall S \subsetneq V, \ |S| \geqslant 2 \end{array} \right\},$$

where $E[S] \subseteq E$ are all edges with both endpoints in S. The polytope $Q \subseteq \mathbb{R}^E$ below describes the natural relaxation of MCCST:

$$Q := \{ x \in P_{\mathrm{ST}} \,|\, a_i \leqslant x(\delta(S_i)) \leqslant b_i \ \forall i \in [k] \} \ .$$

Unfortunately, solutions to the relaxation $\min\{c^\top x \mid x \in Q\}$ are too weak for our purposes. In particular, there are instances where there exists a solution $y \in Q$ fulfilling the chain-constraints, even though any spanning tree must violate at least one chain constraint by a factor of at least 2. (We provide such an example in Appendix A.) Hence, when comparing any integral solution to y, it will be impossible to stay within a factor of $1 + \varepsilon$ regarding the violation of constraints—but this is precisely what we want to achieve. This also shows a hard limit for prior approaches, which are all based on Q.

We therefore aim for a stronger relaxation. It turns out that the reason why Q can be a bad relaxation is the potential existence of small bounds a_i, b_i. Indeed, assume that all a_i, b_i for $i \in [k]$ were at least $c \cdot \log k$ for a sufficiently large constant c (depending on ε). Then one could first find an optimal solution x^* to $\min\{c^\top x \mid x \in Q\}$, and then round x^* to a spanning tree by using one of several negatively correlated rounding procedure developed within the last 10 years (see [2, 8]), which lead to Chernoff-type concentration bounds. The theorem below summarizes a simplified form of the properties obtained by those procedures.[3]

Theorem 5 (see [2, 8]). *Let $y \in P_{\mathrm{ST}}$. There exists an efficient randomized rounding scheme for rounding y to a random spanning tree T in G such that:*

(i) $\Pr[e \in T] = y_e$ *for all $e \in E$, and*
(ii) *For any $\lambda > 0$ and $U \subseteq E$, we have*

$$\Pr\left[(1 - \lambda)y(U) \leqslant |T \cap U| \leqslant (1 + \lambda)y(U)\right] \geqslant 1 - 2e^{-y(U)\lambda^2/3} \ .$$

Consider applying Theorem 5 to x^* to obtain a spanning tree T. By choosing $U = \delta(S_i)$ in the above theorem for any $i \in [k]$, one obtains that $|T \cap \delta(S_i)|$ is within a $(1 \pm \varepsilon)$-factor of $x^*(\delta(S_i))$ with probability $1 - k^{\Omega(1)}$ if $x^*(\delta(S_i)) \geqslant c \cdot \log k$. Moreover, $x^* \in Q$ implies $a_i \leqslant x^*(\delta(S_i)) \leqslant b_i$ for $i \in [k]$. Hence, a union bound over all chain constraints shows that T is unlikely to violate any chain constraint by a large factor.

Motivated by this observation, we design a dynamic programming approach to find points $y \in Q$ of small cost that, for each $i \in [k]$, are either integral on the edges $\delta(S_i)$, or have a large value $y(\delta(S_i))$. To formalize this idea, we introduce the notion of τ-integral solutions.

---

[3]More generally, randomized rounding procedures with these properties can be obtained for any matroid base polytope and Chernoff-type concentration holds for any linear function with small non-negative coefficients (see [8]).

Definition 6 (τ-integral). *For $\tau \in \mathbb{Z}_{\geqslant 0}$, we say that a point $y \in \mathbb{R}^E$ is τ-integral (with respect to the cuts $S_1, \ldots, S_k$), if for each $i \in [k]$, either*

(i) $y(\delta(S_i)) \leqslant \tau$ and y is integral on the edges in $\delta(S_i)$, or
(ii) $y(\delta(S_i)) \geqslant \tau + 1$.

We call the cuts S_i satisfying (i) and (ii), respectively, the y-small and y-large cuts.

Clearly, any integral point is τ-integral for any $\tau \in \mathbb{Z}_{\geqslant 0}$. The key implication of our dynamic programming approach in the context of MCCST is the following.

Theorem 7. *For any $\tau \in \mathbb{Z}_{\geqslant 0}$, there is an algorithm that returns in $|V|^{\mathcal{O}(\tau)}$ time a τ-integral point $y \in Q$ with $c^\top y \leqslant c(\mathrm{OPT})$, where OPT is an optimal solution to MCCST.*

Not surprisingly, to obtain Theorem 7 we want our dynamic program to guess edges in the cuts that will later be y-small. However, this simple high-level plan comes with some important technical hurdles. In particular, even if we knew the edges used in some cut $\delta(S_i)$, completing the two parts of the spanning tree on the left-hand side of the cut (on the vertices S_i) and on its right-hand side (on $V \setminus S_i$), respectively, are two highly dependent subprobems. Interestingly, it is not easy to separate them into independent ones by guessing further structure, like the connectedness on each side, without creating NP-hard subproblems. We expand on these, and further issues, in Section 3, and show how one can address them. A key difference between classical dynamic programs and our approach is that our propagation step requires a fractional solution of a previous subproblem, and not just a small fingerprint of previously obtained solutions.

The issue of small cuts is now resolved by Theorem 7 by setting $\tau = \Theta(k)$ and rounding a τ-integral point $y \in Q$ using a randomized rounding procedure with the guarantees stated in Theorem 5: Because y is integral on y-small cuts, the rounding procedure will return a tree T such that χ^T coincides with y on all y-small cuts, because it is marginal-preserving (property (i) in Theorem 5).

One last technical hurdle to overcome to obtain a $(1, 1 + \varepsilon)$-approximation for MCCST is that the properties of a negatively correlated rounding procedure, as stated in Theorem 5, are not enough to get a spanning tree that both (i) violates chain constraints at most slightly, and (ii) has cost no more than $c^\top y$. Indeed, typical applications of such rounding procedures only lead to $(1 + \varepsilon)$-approximations in terms of the objective (see [8, 7] for examples). We show that this loss in the objective is avoidable in MCCST, and other settings, by using a simple alteration step that modifies the obtained spanning tree by swapping one edge.

Theorem 8. *Let $y \in P_{\mathrm{ST}}$ and $c \in \mathbb{R}^E$. Let T be a random spanning tree in $G = (V, E)$ drawn from a distribution satisfying $\Pr[e \in T] = y(e)$ for all $e \in E$. Let $\overline{T}$ be a spanning tree minimizing $c(U)$ among all spanning trees U whose symmetric difference $U \triangle T := (U \setminus T) \cup (T \setminus U)$ with T satisfies $|U \triangle T| \leqslant 2$ and such that $y(e) \in (0, 1)$ for $e \in U \triangle T$. Then*

$$\Pr\left[c(\overline{T}) \leqslant c^\top y\right] \geqslant (|V| - 1)^{-1} .$$

In Section 4, we show that Theorem 8 holds even in a much more general context and has implications outside MCCST. For the specific setting of MCCST, we observe in Section 4 that also a method introduced by Linhares and Swamy [20] can be adapted to avoid the $(1 + \varepsilon)$-factor loss in the objective.

We can now put together the above ingredients to obtain our quasi-polynomial $(1, 1 + \varepsilon)$-approximation for MCCST, stated as Algorithm 1 below.

Algorithm 1: Quasi-polynomial $(1, 1 + \varepsilon)$-approximation for MCCST
---
1. Let $\tau := \lfloor 96 \log(2|V|)/\varepsilon^2 \rfloor$, and use Theorem 7 to find a τ-integral point $y \in Q$ with $c^\top y \leqslant c(\text{OPT})$.

2. Let $\ell := \lceil 2|V| \log |V| \rceil$, and randomly round y with a rounding procedure as guaranteed by Theorem 5, ℓ times independently, to obtain spanning trees $T_1, \ldots, T_\ell$.

3. For each $j \in [\ell]$, find a minimum cost spanning tree $\overline{T}_j$ among all spanning trees T with $|T \triangle T_j| \leqslant 2$ and such that $y(e) \in (0, 1)$ for all $e \in T \triangle T_j$.

4. Among all $\overline{T}_j$ for $j \in [\ell]$ with $\frac{a_i}{1+\varepsilon} \leqslant |\overline{T}_j \cap \delta(S_i)| \leqslant (1+\varepsilon)b_i$ for all $i \in [k]$, return one of smallest cost.

We now show that the above results—in particular Theorem 7, which follows from our dynamic program, and Theorem 8—imply that Algorithm 1 is a quasi-polynomial $(1, 1 + \varepsilon)$-approximation for MCCST.

Proof of Theorem 1. We will show that with probability at least $1 - 1/|V|$, there is one spanning tree $\overline{T}_j$ among the trees $\overline{T}_1, \ldots, \overline{T}_\ell$ computed by Algorithm 1 that satisfies both

(i) $\frac{1}{1+\varepsilon} \cdot y(\delta(S_i)) \leqslant |\overline{T}_j \cap \delta(S_i)| \leqslant (1+\varepsilon) \cdot y(\delta(S_i))$ for all $i \in [k]$, and

(ii) $c(\overline{T}_j) \leqslant c^\top y$, $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ (1)

which indeed implies that the returned solution is $(1, 1 + \varepsilon)$-approximate because $a_i \leqslant y(\delta(S_i)) \leqslant b_i$ for all $i \in [k]$ due to $y \in Q$, and y satisfies $c^\top y \leqslant c(\text{OPT})$ as guaranteed by step 1 of the algorithm.

We first analyze a single random spanning tree among the spanning trees $T_1, \ldots, T_\ell$ determined in step 2 of the algorithm. We denote by T such a spanning tree that was obtained by randomly rounding y with a randomized rounding procedure as guaranteed by Theorem 5. Observe that because the rounding is marginal-preserving, χ^T coincides with y on any edge $e \in E$ with $y(e) \in \{0, 1\}$. As y is τ-integral, all edges within y-small cuts are of this type and T thus fulfills all chain constraints corresponding to y-small cuts. Together with Chernoff-type concentration bounds guaranteed by Theorem 5, applied with $\lambda = \varepsilon/4$ and using $y(\delta(S_i)) \geqslant \tau + 1$ for y-large cuts, we have

$$\Pr\left[\left(1 - \frac{\varepsilon}{4}\right) y(\delta(S_i)) \leqslant |T \cap \delta(S_i)| \leqslant \left(1 + \frac{\varepsilon}{4}\right) y(\delta(S_i))\right] \geqslant 1 - \frac{1}{2|V|^2} \quad \forall i \in [k] \ . \tag{2}$$

Now let $\overline{T}$ be a spanning tree of minimum cost among all spanning trees $U \subseteq E$ with $|U \triangle T| \leqslant 2$ and $y(e) \in (0, 1)$ for $e \in U \triangle T$. The cost of this spanning tree has the same distribution as the cost of the spanning trees $\overline{T}_1, \ldots, \overline{T}_\ell$ computed in step 3 of Algorithm 1. By Theorem 8, we have

$$\Pr\left[c(\overline{T}) \leqslant c^\top y\right] \geqslant \frac{1}{|V| - 1} \ . \tag{3}$$

Using a union bound over the k events described in (2) and the one described in (3), we obtain that T and $\overline{T}$ simultaneously fulfill

(a) $\left(1 - \frac{\varepsilon}{4}\right) y(\delta(S_i)) \leqslant |T \cap \delta(S_i)| \leqslant \left(1 + \frac{\varepsilon}{4}\right) y(\delta(S_i))$ for all $i \in [k]$, and

(b) $c(\overline{T}) \leqslant c^\top y$,

with probability at least

$$1 - \left(k \cdot \frac{1}{2|V|^2} + \left(1 - \frac{1}{|V| - 1}\right)\right) \geqslant \frac{1}{2|V|} \ ,$$

where we used $k \leqslant |V|$ in the above inequality. Next, we show that property (a) above implies

$$\left(1 - \frac{\varepsilon}{2}\right) y(\delta(S_i)) \leqslant |\overline{T} \cap \delta(S_i)| \leqslant \left(1 + \frac{\varepsilon}{2}\right) y(\delta(S_i)) \quad \forall i \in [k] \ , \tag{4}$$

which in turn implies $\frac{1}{1+\varepsilon} \cdot y(\delta(S_i)) \leqslant |\overline{T} \cap \delta(S_i)| \leqslant (1 + \varepsilon) \cdot y(\delta(S_i))$ for all $i \in [k]$, providing the property that we seek as highlighted in (1). To see that (4) holds for any $i \in [k]$ that corresponds to a y-small cut, notice that for such i we have $\overline{T} \cap \delta(S_i) = T \cap \delta(S_i)$, as $\overline{T}$ and T only differ on edges on which y has a fractional value, and, due to τ-integraliy of y, small cuts do not contain such edges. Hence, consider $i \in [k]$ with $y(\delta(S_i)) \geqslant \tau + 1$. Because $|T \triangle \overline{T}| \leq 2$, $\overline{T}$ is either the same as T or obtained from T by replacing one edge by a different one, so

$$|T \cap \delta(S_i)| - 1 \leqslant |\overline{T} \cap \delta(S_i)| \leqslant |T \cap \delta(S_i)| + 1 \ .$$

The relation (4) for y-large cuts now follows from the above inequality and (a):

$$|\overline{T} \cap \delta(S_i)| \geqslant \left(1 - \frac{\varepsilon}{4}\right) \cdot y(\delta(S_i)) - 1 \geqslant \left(1 - \frac{\varepsilon}{2}\right) \cdot y(\delta(S_i)) \ , \quad \text{and}$$
$$|\overline{T} \cap \delta(S_i)| \leqslant \left(1 + \frac{\varepsilon}{4}\right) \cdot y(\delta(S_i)) + 1 \leqslant \left(1 + \frac{\varepsilon}{2}\right) \cdot y(\delta(S_i)) \ ,$$

where the second inequality in each of the two above lines follows from $y(\delta(S_i)) \geqslant \tau + 1 \geqslant {96 \log(2|V|)}/{\varepsilon^2}$, because S_i is y-large.

In summary, the tree $\overline{T}$ satisfies the two desired properties highlighted in (1) with probability at least $(2|V|)^{-1}$. Because the algorithm computes $\ell = \lceil 2|V| \log |V| \rceil$ independent random trees $\overline{T}_1, \dots, \overline{T}_j$ with the same distribution as $\overline{T}$, the probability that at least one of them fulfills the properties highlighted in (1) is at least

$$1 - \left(1 - \frac{1}{2|V|}\right)^{\ell} \geqslant 1 - e^{-\frac{\ell}{2|V|}} \geqslant 1 - \frac{1}{|V|} \ ,$$

as desired. Finally, the running time is dominated by the quasi-polynomial time dynamic programming approach used to find a cheap τ-integral point $y \in Q$ in step 1 of Algorithm 1. (All other steps of the algorithm can be performed efficiently.) By Theorem 7, we thus get a running time bound $|V|^{\mathcal{O}(\tau)} = |V|^{\mathcal{O}(\log |V|/\varepsilon^2)}$. $\square$

3 The dynamic programming approach for MCCST

First observe that to prove Theorem 7, it suffices to consider $\tau \leqslant |V| - 1$, because any τ-integral point in $y \in Q$ for $\tau \geqslant |V| - 1$ is integral as $Q \subseteq P_{\text{ST}}$, and the y-value on any cut is at most $|V| - 1$ for any point in P_{ST}. Thus, any $\tau \geqslant |V|$ can be replaced by $\tau = |V| - 1$, so we assume $\tau \leqslant |V| - 1$ in what follows.

Our dynamic program to find a cheap τ-integral point in Q is inspired by recent dynamic programming approaches in the context of Path TSP [30, 34], but faces important new technical challenges that require novel conceptual insights. To highlight this point, let us first consider the significantly simpler special case of $\tau = 1$. The dynamic programming approaches for Path TSP are essentially algorithms for this case.[4]

### 3.1 Brief overview to find cheap 1-integral solution following prior techniques

To gain intuition for this special case, which nicely allows for showcasing later on the added difficulty faced for general τ, assume that we knew upfront the small cuts with respect to an optimal solution $\text{OPT} \subseteq E$, i.e., the cuts among $S_1, \dots, S_k$ in which OPT contains a single edge. Let $S_{i_1}, \dots, S_{i_\ell}$ for $1 \leqslant i_1 < \dots < i_\ell \leqslant k$ be these small cuts. For notational convenience, we set $S_{i_0} := \emptyset$ and $S_{i_{\ell+1}} := V$. Now consider the $\ell + 1$ induced subgraphs $G[S_{i_1}], G[S_{i_2} \setminus S_{i_1}], \dots, G[S_{i_\ell} \setminus S_{i_{\ell-1}}], G[V \setminus S_{i_\ell}]$.[5] It is not hard to observe that the edges of OPT within each of these subgraphs must form a spanning tree in that subgraph. Moreover, for

---
[4]More precisely, dynamic programs for Path TSP are looking for points in the Held-Karp relaxation of Path TSP instead of the spanning tree polytope, but this is only a minor technical difference without significant impact on the dynamic program.

[5]For $W \subseteq V$, $G[W]$ is the subgraph of G induced by W.

$j \in [\ell]$, the single edge $e_j \in \mathrm{OPT} \cap \delta(S_{i_j})$ must go from $S_{i_j} \setminus S_{i_{j-1}}$ to $S_{i_{j+1}} \setminus S_{i_j}$ for OPT to be a spanning tree (see Figure 1a). If, moreover, we even knew the single edge $e_j \in \delta(S_{i_j}) \cap \mathrm{OPT}$ for each $j \in [\ell]$, then the problem of finding a *corresponding* cheapest 1-integral point $y \in P_{\mathrm{ST}}$—i.e., with y-small cuts $S_{i_1}, \ldots, S_{i_\ell}$ and edges $e_1, \ldots, e_\ell$ contained in them—decomposes into $\ell + 1$ independent linear programs, one within each of the above-mentioned induced subgraphs. More precisely, one has to find, for $j \in [\ell + 1]$, a cheapest point y^j in the spanning tree polytope of $G[S_{i_j} \setminus S_{i_{j-1}}]$ with lower bounds on each cut S_i with $S_{i_j} \subsetneq S_i \subsetneq S_{i_{j-1}}$ to make sure that y^j, together with the guessed edges in small cuts, has a load of at least 2 on these cuts.

The above observations now naturally lead to a dynamic programming approach that extends solutions from left to right, i.e., a 1-integral solution in some subgraph $G[S_i]$ for some i is extended to one on $G[S_j]$ for $j > i$. This way, one can use a dynamic program to optimize over all possibilities of small cuts and edges contained in them (see [30, 34] for more details of this approach in the context of Path TSP).


(a) Independent problems.   **(b)** Dependent problems.
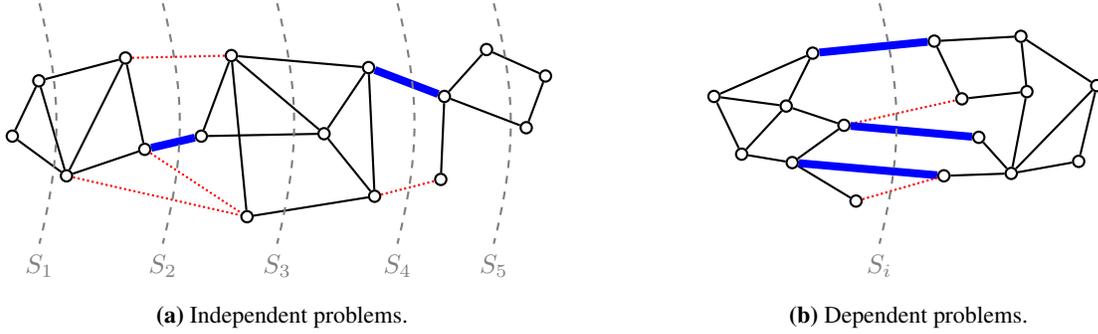
Figure 1: (a) Here, the small cuts are $S_{i_1} = S_2$ and $S_{i_2} = S_4$, and we assume that only a single edge, drawn in thick and blue, crosses each small cut. Finding a τ-integral point $y \in P_{\mathrm{ST}}$ boils down to solving independent subproblems in $G[S_2]$, $G[S_4 \setminus S_2]$, and $G[V \setminus S_4]$. **(b)** Because more than one edge is in the small cut S_i, the problem does not decompose into independent subproblems on the left-hand side and right-hand side of S_i.

3.2 Toward general τ with connectivity patterns and resulting challenges

However, if $\tau \geqslant 2$, i.e., if there are two or more edges in small cuts, splitting the problem into independent ones along small cuts comes with significant additional challenges linked to obtaining connectivity and acyclicity globally from independent solutions of the subproblems (see Figure 1b).

One natural approach to try to address such challenges is to maintain more structure in the dynamic program, by for example also enumerating over potential *connectivity patterns* of edges in small cuts, i.e., ways of how the edges in a small cut could be connected on one or either side of the cut. For ease of presentation, consider a situation with a single small cut S_i, and a given selection of $t \leqslant \tau$ many edges $F := \{\{u_j, v_j\} \mid j \in [t]\} \subseteq \delta(S_i)$ with $u_j \in S_i$ and $v_j \notin S_i$ for all $j \in [t]$. A *connectivity pattern* for the right-hand side of the cut is a partition $\mathcal{C}$ of $\{v_1, \ldots, v_t\}$, where a set $C \in \mathcal{C}$ indicates that the vertices in C shall be connected in $G[V \setminus S_i]$. We call the triple $(S_i, F, \mathcal{C})$ a *connectivity triple* (see Figure 2).
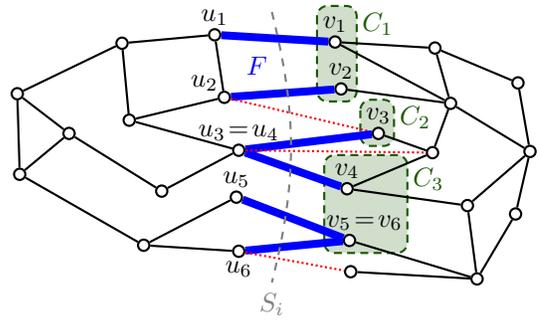
Figure 2: A connectivity triple $(S_i, F, \mathcal{C})$ with connectivity pattern $\mathcal{C} = \{C_1, C_2, C_3\}$.

Definition 9 (Compatibility with a connectivity triple)**.**

(i) *A spanning tree $T \subseteq E$ is* compatible *with the connectivity triple $(S_i, F, \mathcal{C})$ if $T \cap \delta(S_i) = F$ and the partition on $\{v_1, \ldots, v_t\}$ induced by the connected components of $T \cap E[V \setminus S_i]$ equals $\mathcal{C}$.*

(ii) *A set $R \subseteq \binom{V \setminus S_i}{2}$ is* right-compatible *with $(S_i, F, \mathcal{C})$ if R is a forest and the partition on $\{v_1, \ldots, v_t\}$ induced by the connected components of R equals $\mathcal{C}$.*

(iii) *Let $U \subseteq E[S_i]$, and let $R \subseteq \binom{V \setminus S_i}{2}$ be right-compatible with $(S_i, F, \mathcal{C})$. Then U is* left-compatible *with $(S_i, F, \mathcal{C})$ if $U \cup F \cup R$ is a spanning tree.*

(iv) *Let $x \in \mathbb{R}^E$ with $\mathrm{supp}(x) \subseteq E[S_i]$, and let $R \subseteq \binom{V \setminus S_i}{2}$ be right-compatible with $(S_i, F, \mathcal{C})$. Then x is* left-compatible *with $(S_i, F, \mathcal{C})$ if $x + \chi^F + \chi^R$ is in the spanning tree polytope of $(V, E \cup R)$.*

We highlight that right-compatible sets R are not required to be a subset of the edges of G, but can contain any pairs of vertices within $V \setminus S_i$. This makes sure that right-compatible sets exist for any connectivity triple, which simplifies the exposition. Moreover, one can observe that the above definitions of left-compatibility do not depend on which right-compatible set R is chosen, and are thus well-defined.

Knowing the correct connectivity triple, the desired separation into independent subproblems can actually be achieved. However, this comes at the cost that the subproblem on the side where we guessed the connectivity pattern becomes substantially harder than in the simple case $\tau = 1$. This is nicely highlighted by a simple connectivity pattern $\mathcal{C}$: Assume that all right endpoints $\{v_1, \ldots, v_t\}$ of F are distinct, t is even, and let $\mathcal{C} = \{\{v_1, v_2\}, \{v_3, v_4\}, \ldots, \{v_{t-1}, v_t\}\}$ be a grouping of the endpoints into pairs. To pinpoint the difficulties, consider the question of whether there exists a right-compatible edge set $U \subseteq E[V \setminus S_i]$. For this to be the case, U must be a forest with $t/2$ components, one for each pair v_j, v_{j+1} that connects that pair. Such a set U exists if and only if there are t vertex-disjoint paths in $G[V \setminus S_i]$, one between v_j and v_{j+1} for each $j \in \{1, 3, \ldots, t-1\}$. Hence, just determining whether there exists a right-compatible solution is at least as difficult as the vertex-disjoint paths problem. Though this problem is efficiently solvable for a constant number of paths $t/2$, due to the seminal results by Robertson and Seymour in the context of the Graph Minor Project (see also [16] for a faster procedure), the known techniques for disjoint paths are highly non-trivial, non-polyhedral, cannot handle costs, and, last but not least, the connectivity pattern $\mathcal{C}$ leading to the disjoint paths problem remains a very special case of connectivity patterns we have to deal with.

3.3 Efficiently extending subsolutions through relaxed connectivity requirements

To overcome this issue in our approach, we still enumerate over connectivity triples with connectivity patterns on the right-hand side of small cuts as described above, but will later not require that right-hand side solutions are right-compatible with it; they only have to properly complete an existing left-hand side solution. The left-hand side solutions, however, will be left-compatible with the guessed connectivity triples.

We start by observing a simple way to describe left-compatibility. Let $(S_i, F, \mathcal{C})$ be a connectivity triple. Then, left-compatible edge sets $U \subseteq E[S_i]$ are simply spanning trees in an auxiliary graph $G(S_i, F, \mathcal{C})$ which we obtain from $G[S_i]$ through the following operations: (i) add the edges F and their endpoints to $G[S_i]$, (ii) contract the vertex sets in $\mathcal{C}$, and (iii) contract the edges in F. There is a canonical one-to-one relation between the edges in $G(S_i, F, \mathcal{C})$ and $E[S_i]$, and we therefore treat them as the same edge set. By $P_{\mathrm{ST}}(S_i, F, \mathcal{C})$, we denote the spanning tree polytope of $G(S_i, F, \mathcal{C})$. Hence, $P_{\mathrm{ST}}(S_i, F, \mathcal{C})$ are all points that are left-compatible with $(S_i, F, \mathcal{C})$.

We are now ready to describe our dynamic programming approach. For ease of notation, we set $S_0 := \emptyset$ and $S_{k+1} := V$. Consider the set $\mathcal{K}$ of all connectivity triples $(S_i, F, \mathcal{C})$, where $i \in \{0, \ldots, k+1\}$ and the crossing edges $F \subseteq \delta(S_i)$ satisfy $a_i \leqslant |F| \leqslant \min\{\tau, b_i\}$. For each $(S_i, F, \mathcal{C}) \in \mathcal{K}$ we determine via our dynamic program a point (partial solution) $y_{(S_i, F, \mathcal{C})} \in \mathbb{R}^{E[S_i]}$ with the following property.

Property 10.

(i) $y_{(S_i, F, \mathcal{C})} \in P_{\mathrm{ST}}(S_i, F, \mathcal{C})$.

(ii) $y_{(S_i, F, \mathcal{C})} + \chi^F$ is τ-integral on $S_1, \ldots, S_{i-1}$.

(iii) $a_h \leqslant y_{(S_i,F,\mathcal{C})}(\delta(S_h)) + |F \cap \delta(S_h)| \leqslant b_h$ *for all* $h \in [i-1]$.

(iv) $c^\top y_{(S_i,F,\mathcal{C})}$ *is at most the cost of a cheapest edge set* $U \subseteq E[S_i]$ *such that* χ^U *fulfills (i), (ii), and (iii).*

For simplicity of notation, we consider all vectors defined on some subset $U \subseteq E$ of the edges, like $y_{(S_i,F,\mathcal{C})}$ above, to be vectors in $\mathbb{R}^E$, where all entries on which the vector was not defined are set to 0.[6] Clearly, if we can obtain such points in $|V|^{\mathcal{O}(\tau)}$ time, then we are done because $y := y_{(V,\emptyset,\{\emptyset\})}$ is in Q because of (i) and (iii); y is a τ-integral point due to (ii); and y satisfies $c^\top y \leqslant c(\mathrm{OPT})$ due to (iv), as desired.

To construct the vectors $y_{(S_i,F,\mathcal{C})}$, we initialize $y_{(S_0,\emptyset,\{\emptyset\})}$ to the zero vector, and consider triples $(S_i, F, \mathcal{C}) \in \mathcal{K}$ in increasing order of i. Let $i \in [k+1]$, let $(S_i, \overline{F}, \overline{\mathcal{C}}) \in \mathcal{K}$, and assume that for each $(S_j, F, \mathcal{C}) \in \mathcal{K}$ with $j < i$, we already computed a vector $y_{(S_j,F,\mathcal{C})}$ satisfying Property 10. To compute a vector $y_{(S_i,\overline{F},\overline{\mathcal{C}})}$ satisfying Property 10, we consider all $(S_j, F, \mathcal{C}) \in \mathcal{K}$ with $j < i$ and for each such triple, we solve the following linear program, which finds a cheapest extension z of $y_{(S_j,F,\mathcal{C})}$ that is left-compatible with $(S_i, \overline{F}, \overline{\mathcal{C}})$:

$$
\begin{aligned}
\min \quad & c^\top z \\
& z \in P_{\mathrm{ST}}(S_i, \overline{F}, \overline{\mathcal{C}}) \\
\max\{\tau + 1, a_h\} \leqslant \; & z(\delta(S_h)) + |\overline{F} \cap \delta(S_h)| \leqslant b_h \quad \forall h \in \{j+1, \dots, i-1\} \\
& z(e) = y_{(S_j,F,\mathcal{C})}(e) \quad\quad\;\; \forall e \in E[S_j] \\
& z(e) = 1 \quad\quad\quad\quad\quad\;\;\; \forall e \in F \\
& z(e) = 0 \quad\quad\quad\quad\quad\;\;\; \forall e \in \delta(S_j) \setminus F \;.
\end{aligned}
\tag{exLP}
$$

Among all linear programs of type (exLP), i.e., one for each triple $(S_j, F, \mathcal{C})$ with $j < i$, we determine the one achieving the smallest optimal value and set $y_{(S_j,\overline{F},\overline{\mathcal{C}})}$ to be an optimal solution of that linear program.[7] This finishes the description of our dynamic program. The bottleneck of the running time is the repeated solving of linear programs of type (exLP). A simple bound on the number of such LPs that we solve is $|\mathcal{K}|^2$, and the running time of $|V|^{\mathcal{O}(\tau)}$ then follows from the following bound and the fact that we can solve (exLP) in strongly polynomial time through standard techniques. A more formal treatment, including a proof of the simple statement below, is given in Section 3.4.

Proposition 11. $|\mathcal{K}| = |V|^{\mathcal{O}(\tau)}$.

We now highlight a few key aspects of our dynamic program. First, even though (exLP) seeks to complete a prior solution $y_{(S_j,F,\mathcal{C})}$ to one for the triple $(S_i, \overline{F}, \overline{C})$, we do not require the completion to be right-compatible with $(S_j, F, \mathcal{C})$.[8] This connectivity pattern is completely disregarded in (exLP) and was only used on the left-hand side of S_j to construct $y_{(S_j,F,\mathcal{C})}$. The key observation is that any integral solution that is compatible with both triples $(S_j, F, \mathcal{C})$ and $(S_i, \overline{F}, \overline{\mathcal{C}})$, and has only large cuts between S_i and S_j, provides a legal way to complete $y_{(S_j,F,\mathcal{C})}$ as formally described by the following statement.

Lemma 12. *Let* $R \subseteq \binom{V \setminus S_i}{2}$ *be right-compatible with* $(S_i, \overline{F}, \overline{\mathcal{C}})$. *Then for any* $U \subseteq E[S_i]$ *such that* $T := U \cup \overline{F} \cup R$ *is a spanning tree compatible with both* $(S_j, F, \mathcal{C})$ *and* $(S_i, \overline{F}, \overline{\mathcal{C}})$, *and* $|T \cap \delta(S_h)| \in [\max\{\tau + 1, a_h\}, b_h]$ *for all* $h \in \{j+1, \dots, i-1\}$, *the following vector is a feasible solution to* (exLP):

$$
z := y_{(S_j,F,\mathcal{C})} + \chi^{F \setminus \overline{F}} + \chi^{U \cap E[S_i \setminus S_j]} \;.
$$

---

[6]This makes sure that expressions like $c^\top y_{(S_i,F,\mathcal{C})}$ or $y_{(S_i,F,\mathcal{C})}(\delta(S_h))$ are well-defined.

[7]We use the usual convention that if some linear program (exLP) is infeasible, then its objective value is interpreted as ∞ (and we will never use a solution to an infeasible linear program later on). Infeasibility occurs, for example, for choices of $(S_j, F, \mathcal{C})$ where F does not contain all edges of $\overline{F} \cap \delta(S_j)$.

[8]Notice that we did not formally define right-compatibility for fractional points because we do not need it, but a natural extension would be to say that it is a convex combination of right-compatible integral solutions.

The above lemma is crucial to make sure that our dynamic program remains a relaxation of the original MCCST problem, even though it requires to complement a specific previously computed solution $y_{(S_j,F,\mathcal{C})}$. Simply speaking, Lemma 12 guarantees that the increment in cost when extending the solution $y_{(S_j,F,\mathcal{C})}$ up to the cut S_i through our dynamic program is no more than the best integral extension that realizes both connectivity triples at S_j and S_i. This makes sure that the solutions we compute fulfill point (iv) of Property 10. Intuitively, the reason why Lemma 12 holds is the following: No matter what precise point $y_{(S_j,F,\mathcal{C})}$ we computed, as long as it is left-compatible with $(S_j, F, \mathcal{C})$, it can be completed by any edge set that is right-compatible with $(S_j, F, \mathcal{C})$ to obtain a point in P_{ST}.

Also note that a crucial difference between our dynamic program and the classical way of using dynamic programming approaches is that we need an *explicit* solution $y_{(S_j,F,\mathcal{C})}$ in our propagation/extension step. Only knowing the connectivity triple $(S_j, F, \mathcal{C}) \in \mathcal{K}$ and the value of a best point $y_{(S_j,F,\mathcal{C})}$ for that triple would not be enough. To highlight this contrast, consider for example a classical dynamic programming approach for the (integer) knapsack problem (see, e.g., [19, Section 17.2]). Here, the dynamic program computes for every possible cost the smallest total weight of items realizing that cost. (Sometimes, the dynamic program is presented with the roles of costs and weights exchanged, in which case, for every possible weight, the dynamic program computes a minimum cost solution of that weight.) In the propagation step of this classical dynamic program, to extend existing partial solutions of smaller cost to partial solutions of larger cost, one only needs to know the weight and cost of previously computed partial solutions, but not the exact partial solution. Partial solutions are sometimes saved in classical dynamic programs to quickly retrieve a solution through backtracking, once the whole dynamic programming table is filled. However, to just determine the optimal value of a solution, and in particular to perform the propagation steps in the dynamic program, partial solutions are not used.

3.4 Details of the dynamic programming approach for MCCST

In this section, we complete the proofs that are missing to formally ensure that the dynamic programming approach described above achieves the guarantees claimed by Theorem 7. Recall that the dynamic program initializes $y_{(S_0,\emptyset,\emptyset)} = 0$, and propagates to points $y_{(S_i,F,\mathcal{C})} \in \mathbb{R}^E_{\geq 0}$ for all $(S_i, F, \mathcal{C}) \in \mathcal{K}$ using Algorithm 2.

Algorithm 2: Propagation to $y_{(S_i,\overline{F},\overline{\mathcal{C}})}$ from all $y_{(S_j,F,\mathcal{C})}$ with $j < i$.

---

1. For all $(S_j, F, \mathcal{C}) \in \mathcal{K}$ with $j < i$, solve the linear program (exLP) and obtain an optimal solution y.

2. Among all solutions y found in step 1, let $y_{(S_i,\overline{F},\overline{\mathcal{C}})}$ be one minimizing $c^\top y$. Return $y_{(S_i,\overline{F},\overline{\mathcal{C}})}$.

We start by proving Lemma 12, which essentially shows that the extension found by our dynamic program when solving a linear program of the form (exLP) has cost no more than the best integral extension that is compatible with the connectivity triples on both sides of the extension.

Proof of Lemma 12. To obtain feasibility of z for (exLP), it is easy to see that the inequality constraints that are stated explicitly in (exLP) follow immediately by definition of z. Thus, it remains to prove that $z \in P_{\mathrm{ST}}(S_i, \overline{F}, \overline{\mathcal{C}})$, i.e., that z is left-compatible with $(S_i, \overline{F}, \overline{\mathcal{C}})$.

By assumption, the tree T is compatible with $(S_j, F, \mathcal{C})$, hence $T \cap \binom{V \setminus S_j}{2}$ is right-compatible with $(S_j, F, \mathcal{C})$. Moreover, $y_{(S_j,F,\mathcal{C})}$ is left-compatible with $(S_j, F, \mathcal{C})$. Combining these observations, we see that $y_{(S_j,F,\mathcal{C})} + \chi^F + \chi^{T \cap \binom{V \setminus S_j}{2}}$ is in the spanning tree polytope of $(V, E \cup R)$. By partitioning $T \cap \binom{V \setminus S_j}{2}$ into $T \cap E[S_i \setminus S_j]$, $\overline{F} \setminus F$ and R, we get that, equivalently, $y_{(S_j,F,\mathcal{C})} + \chi^F + \chi^{T \cap E[V \setminus S_j]} + \chi^{\overline{F} \setminus F} + \chi^R$ is in the spanning tree polytope of $(V, E \cup R)$, which implies that $y_{(S_j,F,\mathcal{C})} + \chi^{F \setminus \overline{F}} + \chi^{T \cap E[S_i \setminus S_j]}$ is left-compatible with $(S_i, \overline{F}, \overline{\mathcal{C}})$. As $T \cap E[S_i \setminus S_j] = U \cap E[S_i \setminus S_j]$, the Lemma follows. $\qquad\square$

With the bound on the incremental cost of an extension from Lemma 12, we can show that propagating along Algorithm 2, we maintain Property 10.

Lemma 13. *Let $i \in [k+1]$ and $(S_i, \overline{F}, \overline{C}) \in \mathcal{K}$. Assume that for all $(S_j, F, C) \in \mathcal{K}$ with $j < i$, we are given points $y_{(S_j, F, C)}$ satisfying Property 10. Then $y_{(S_i, \overline{F}, \overline{C})}$ obtained by Algorithm 2 also has Property 10.*

Proof. Note that any solution of (exLP) satisfies points (i), (ii) and (iii) of Property 10 with respect to $(S_i, \overline{F}, \overline{C})$. Point (i) follows directly from the corresponding constraint in (exLP), while (ii) and (iii) are implied by the constraints in the linear program for cuts S_h with $h \geqslant j$, and follow for cuts S_h with $h < j$ from the fact that $y_{(S_j, F, C)}$ has Property 10, by assumption.

To see that point (iv) holds, it is enough to see that for any edge set $U \subseteq E[S_i]$ fulfilling (i), (ii) and (iii), there is a connectivity triple (S_i, F, C) such that the corresponding solution y of the linear program (exLP) satisfies $c^\top y \leqslant c(U)$. To see this, fix such an edge set U, which is by definition left-compatible with $(S_i, \overline{F}, \overline{C})$. Thus, for any $R \subseteq \binom{V \setminus S_i}{2}$ that is right-compatible with $(S_i, \overline{F}, \overline{C})$, the edge set $T := U \cup \overline{F} \cup R$ is a spanning tree in $(V, E \cup R)$ compatible with $(S_i, \overline{F}, \overline{C})$. Let $j < i$ be maximal such that S_j is a T-small cut. Let $F = T \cap \delta(S_j)$ and let C be the connectivity pattern such that T is compatible with (S_j, F, C).

We claim that (S_j, F, C) is the connectivity triple that we are looking for, i.e., if y is an optimal solution of (exLP) when extending $y_{(S_j, F, C)}$ to $(S_i, \overline{F}, \overline{C})$, then $c^\top y \leqslant c(U)$. Thereto, observe that by definition, $T = U \cup \overline{F} \cup R$ is a spanning tree in $(V, E \cup R)$ compatible with both (S_j, F, C) and $(S_i, \overline{F}, \overline{C})$, and by the choice of j, we also have $|T \cap \delta(S_h)| \in [\max\{\tau + 1, a_h\}, b_h]$ for all $h \in \{j+1, \ldots, i-1\}$. Thus, R and U satisfy the assumptions of Lemma 12, and we get that $z := y_{(S_j, F, C)} + \chi^{F \setminus \overline{F}} + \chi^{U \cap E[S_i \setminus S_j]}$ is a feasible solution to (exLP). But y is an optimal solution of the same linear program, hence

$$c^\top y \leqslant c^\top y_{(S_j, F, C)} + c(\chi^{F \setminus \overline{F}}) + c(U \cap E[S_i \setminus S_j]) \ .$$

By assumption, $y_{(S_j, F, C)}$ satisfies Property 10, and applying point (iv) with the edge set $U \cap E[S_j]$, we get $c^\top y_{(S_j, F, C)} \leqslant c(U \cap E[S_j])$. Combining this with the above, we obtain the desired inequality

$$c^\top y \leqslant c(U \cap E[S_j]) + c(\chi^{F \setminus \overline{F}}) + c(U \cap E[S_i \setminus S_j]) = c(U) \ . \qquad \square$$

The final step before wrapping up and proving Theorem 7 is to show the bound on $|\mathcal{K}|$ from Proposition 11.

Proof of Proposition 11. The set $\mathcal{K}$ consists of all connectivity triples (S_i, F, C), where $i \in \{0, 1, \ldots, k+1\}$, $F \subseteq \delta(S_i)$ has size at most τ, and C is a corresponding connectivity pattern. In order to build such a triple, there are at most $|V| + 1$ options for choosing a set S_i (one for each $i \in \{0, \ldots, k+1\}$, and we have $k \leqslant |V| - 1$). Furthermore, note that $\delta(S_i)$ can contain at most $\mathcal{O}(|V|^2)$ many edges of G, hence there are at most $|V|^{\mathcal{O}(\tau)}$ many choices for a subset of size at most τ. Finally, C is a partition of the at most τ many endpoints of the edges in F that do not lie inside S_i, and the number of such partitions can be bounded by $|V|^{\mathcal{O}(\tau)}$ as well, where we use $\tau \leqslant |V| - 1$. In total, we thus get $|\mathcal{K}| \leqslant |V|^{\mathcal{O}(\tau)}$. $\square$

Finally, we are ready to prove Theorem 7.

Proof of Theorem 7. We calculate points $y_{(S_i, F, C)}$ for all $(S_i, F, C) \in \mathcal{K}$ by initializing $y_{(\emptyset, \emptyset, \{\emptyset\})} = 0$ and calculating $y_{(S_i, F, C)}$ in increasing order of i using the propagation step described in Algorithm 2. Note that $y_{(\emptyset, \emptyset, \{\emptyset\})}$ has Property 10, and hence, by an inductive application of Lemma 13, we obtain that all points $y_{(S_i, F, C)}$ for $(S_i, F, C) \in \mathcal{K}$ have Property 10. In particular, Property 10 for $y_{(V, \emptyset, \{\emptyset\})}$ immediately implies that this point has the properties claimed by Theorem 7. Note that the guarantee on $c^\top y_{(V, \emptyset, \{\emptyset\})}$ follows from the fact that a cheapest edge set U such that χ^U fulfils points (i), (ii) and (iii) of Property 10 with respect to $(S_i, F, C) = (V, \emptyset, \{\emptyset\})$ is in fact an optimal solution.

For the running time bound, observe that the dominating operation of our dynamic programming procedure is repeatedly solving linear programs of type (exLP). The total number of such linear programs that we solve is bounded from above by $|\mathcal{K}|^2$, and therefore, the running time of $|V|^{\mathcal{O}(\tau)}$ follows from Proposition 11 and the observation that linear programs of the type (exLP) can be solved in strongly polynomial time. The latter can be achieved by using a compact extended formulation for the spanning tree polytope with small coefficients in the constraint matrix (one can, for example, use the one by Martin [21], which has coefficients that are bounded by 1 in absolute value), and then applying the framework of Tardos [29]. $\qquad\square$

4   Local correction steps for rounding procedures in $\{0, 1\}$-polytopes

In this section, we discuss details on the proof of Theorem 8, used to avoid a $(1 + \varepsilon)$-factor loss in the objective value. We present this result separately because it may be of independent interest, as it applies to a broad class of problem settings. At the end of this section, in Section 4.3, we briefly discuss how, for MCCST, an alternative approach introduced by Linhares and Swamy [20] also allows for avoiding a loss in the objective value.

We show Theorem 8 by proving a more general statement for $\{0, 1\}$-polytopes, based on polyhedral neighborhoods. We therefore start with some basic polyhedral terminology.[9] $\{0, 1\}$-polytopes are a representation of *set systems* $(E, \mathcal{F})$, where E is a finite ground set and $\mathcal{F} \subseteq 2^E$. One can think of $\mathcal{F}$ as the feasible sets of some combinatorial problem over E. For example, E may be the edge set of a graph and $\mathcal{F}$ the family of all spanning trees. The *combinatorial polytope* $P_\mathcal{F}$ that corresponds to $\mathcal{F}$ is defined by

$$P_\mathcal{F} := \mathrm{conv}(\{\chi^F \mid F \in \mathcal{F}\}) \ , \tag{5}$$

where conv denotes the convex hull. Hence, if $\mathcal{F}$ are all spanning trees of a graph, $P_\mathcal{F}$ is the spanning tree polytope. For $F_1, F_2 \in \mathcal{F}$ and $q \in \mathbb{Z}_{\geqslant 0}$, we say that F_2 is in the q-neighborhood of F_1 on $P_\mathcal{F}$ if one can reach the vertex χ^{F_2} of $P_\mathcal{F}$ from the vertex χ^{F_1} by successively traversing at most q edges of $P_\mathcal{F}$. Notice that this natural notion extends the way we modify T to obtain $\overline{T}$ in Theorem 8: Indeed, this follows from the well-known property that two spanning trees $T, \overline{T}$ in G—or, more generally, any two bases $T, \overline{T}$ of a matroid—have the property that $\chi^T, \chi^{\overline{T}}$ are adjacent in $P_\mathcal{F}$ if and only if $|T \triangle \overline{T}| = 2$ (see, e.g., [24, Volume B]). Furthermore, for any $y \in P_\mathcal{F}$, we denote by $P_{\mathcal{F}_y} \subseteq P_\mathcal{F}$ the minimal face of $P_\mathcal{F}$ that contains y. Additionally, $\mathcal{F}_y \subseteq \mathcal{F}$ denotes all sets $F \in \mathcal{F}$ such that $\chi^F \in P_{\mathcal{F}_y}$. (Note that these definitions of $\mathcal{F}_y$ and $P_{\mathcal{F}_y}$ are consistent with (5) in the sense that $P_{\mathcal{F}_y}$ is indeed the combinatorial polytope of the family $\mathcal{F}_y$.)

A key quantity in our derivations is the cardinality $\rho(\mathcal{F})$ of a largest size set in $\mathcal{F}$:

$$\rho(\mathcal{F}) := \max\{|F| \mid F \in \mathcal{F}\} \ .$$

Typically, when having a set system $(E, \mathcal{F})$ where all sets have the same cardinality, i.e., $|F| = \rho(\mathcal{F})$ for all $F \in \mathcal{F}$, we can obtain slightly stronger results later. We call such set systems *equal-cardinality* systems. Note that the family of spanning trees (or bases of any matroid) is an equal-cardinality system. We prove the following generalization of Theorem 8.

Theorem 14. *Let $(E, \mathcal{F})$ be a set system, let $y \in P_\mathcal{F}$, $c \in \mathbb{R}^E$, $q \in \mathbb{Z}_{\geqslant 1}$, and let T be a random set in $\mathcal{F}$ drawn from a distribution satisfying $\Pr[e \in T] = y_e$ for all $e \in E$. Let $\overline{T} \in \mathcal{F}$ be a set minimizing $c(U)$ among all $U \in \mathcal{F}$ in the q-neighborhood of T on $P_{\mathcal{F}_y}$. Then*

$$\Pr\left[c(\overline{T}) \leqslant c^\top y\right] \geqslant \frac{q}{2\rho(\mathcal{F})} \ .$$

[9]We refer the interested reader to [24, Volume A] for more information on polyhedral combinatorics.

Moreover, if $(E, \mathcal{F})$ *is an equal-cardinality system, then*

$$\Pr\left[c(\overline{T}) \leqslant c^\top y\right] \geqslant \frac{q}{\rho(\mathcal{F})} \quad .$$

First observe that Theorem 14 indeed implies Theorem 8.

Proof of Theorem 8. We set E to be the edges of $G = (V, E)$, $\mathcal{F} \subseteq 2^E$ to be all spanning trees in G, and $q = 1$. Clearly, in this case we have $\rho(\mathcal{F}) = |V| - 1$ and $|F| = \rho(\mathcal{F})$ for all $F \in \mathcal{F}$, because every spanning tree has precisely $|V| - 1$ edges. Hence spanning trees form an equal-cardinality system. By Theorem 14 we thus obtain

$$\Pr[c(W) \leqslant c^\top y] \geqslant \frac{1}{|V| - 1} \quad ,$$

where $W \in \mathcal{F}$ is a set minimizing $c(U)$ among all $U \subseteq \mathcal{F}$ in the 1-neighborhood of T on $P_{\mathcal{F}_y}$.

For the above to imply Theorem 8, it suffices to show that any $U \subseteq \mathcal{F}$ in the 1-neighborhood of T on $P_{\mathcal{F}_y}$ fulfills $y(e) \in (0, 1)$ for all $e \in U \Delta T$. Because $P_{\mathcal{F}} \subseteq [0, 1]^E$—i.e., non-negativity constraints and constraints of type $x(e) \leqslant 1$ are valid for $P_{\mathcal{F}}$—all points on $P_{\mathcal{F}_y}$ coincide with y on the edges where y is integral, i.e.,

$$P_{\mathcal{F}_y} \subseteq \left\{ x \in [0, 1]^E \mid x(e) = y(e) \; \forall e \in E \text{ with } y(e) \in \{0, 1\} \right\} \quad .$$

This implies that any $F \subseteq \mathcal{F}_y$ fulfills $y(e) \in (0, 1)$ for all $e \in F \triangle T$. Hence, this also holds for any $U \subseteq \mathcal{F}$ in the 1-neighborhood of T on $P_{\mathcal{F}_y}$, as desired, and finishes the proof. $\qquad\square$

4.1 Proof of Theorem 14

We show Theorem 14 in several steps. We first derive a bound on the cost of a well-chosen set A in the 1-neighborhood of another fixed set F. The following lemma formalizes this statement. The cost improvement is measured with respect to the cost of some target set Q, which will later be chosen to be a set in $\mathcal{F}$ of smallest cost.

Lemma 15. *Let* $(E, \mathcal{F})$ *be a set system. Let* $c \in \mathbb{R}^E$, *and* $F, Q \in \mathcal{F}$ *with* $F \neq Q$. *Then there exists a neighbor* $A \in \mathcal{F}$ *of* F *on* $P_{\mathcal{F}}$ *satisfying*

- (i) $c(A) - c(Q) \leqslant \left(1 - \frac{1}{|Q \triangle F|}\right) \cdot (c(F) - c(Q))$, *and*
- (ii) $|Q \triangle A| \leqslant |Q \triangle F| - 1$.

Moreover, if $(E, \mathcal{F})$ *is an equal-cardinality system, then the above properties can be strengthened to*

- (i') $c(A) - c(Q) \leqslant \left(1 - \frac{2}{|Q \triangle F|}\right) \cdot (c(F) - c(Q))$, *and*
- (ii') $|Q \triangle A| \leqslant |Q \triangle F| - 2$.

Proof. Consider the vertex χ^F of $P_{\mathcal{F}}$, and the family $\{A_1, \ldots, A_\ell\} \in \mathcal{F}$ of all neighboring sets in $\mathcal{F}$ on $P_{\mathcal{F}}$. We start with a basic polyhedral property, namely that the cone with apex χ^F spanned by all edges of $P_{\mathcal{F}}$ incident with χ^F contains the whole polytope, i.e.,

$$P_{\mathcal{F}} \subseteq \chi^F + \text{cone}\left(\{\chi^{A_i} - \chi^F \mid i \in [\ell]\}\right) \quad .$$

In particular, this implies that there exist coefficients $\lambda_i \geqslant 0$ for $i \in [\ell]$ such that

$$\chi^Q = \chi^F + \sum_{i=1}^{\ell} \lambda_i \cdot \left(\chi^{A_i} - \chi^F\right) \quad .$$

We are only interested in strictly positive coefficients. Let k be the number of strictly positive coefficients, and assume, by renumbering the indices, that these are the coefficients $\lambda_1, \ldots, \lambda_k$. Hence,

$$\chi^Q = \chi^F + \sum_{i=1}^{k} \lambda_i \cdot \left(\chi^{A_i} - \chi^F \right) \quad , \tag{6}$$

and $\lambda_i > 0$ for all $i \in [k]$. Let $\lambda := \sum_{i=1}^{k} \lambda_i$.

Claim. We have $\lambda \leqslant |Q \,\triangle\, F|$. Moreover, if $(E, \mathcal{F})$ is an equal-cardinality system, then $\lambda \leqslant \frac{1}{2} |Q \,\triangle\, F|$.

Proof of claim. We have

$$
\begin{aligned}
|Q \Delta F| &= \| \chi^Q - \chi^F \|_1 \\
&= \left\| \sum_{i=1}^{k} \lambda_i \cdot \left(\chi^{A_i} - \chi^F \right) \right\|_1 \\
&= \left\| \sum_{i=1}^{k} \lambda_i \cdot \left(\chi^{A_i \setminus F} - \chi^{F \setminus A_i} \right) \right\|_1 \\
&= \sum_{i=1}^{k} \lambda_i \cdot \left\| \chi^{A_i \setminus F} - \chi^{F \setminus A_i} \right\|_1 \\
&= \sum_{i=1}^{k} \lambda_i \cdot |A_i \,\triangle\, F| \quad , \tag{7}
\end{aligned}
$$

where the second equality follows from (6), and the forth one from the fact that all vectors $\chi^{A_i \setminus F} - \chi^{F \setminus A_i}$ in the sum are non-positive on entries corresponding to F and non-negative on all other entries. Now, because A_i are neighbors of F on $P_{\mathcal{F}}$, we have $A_i \neq F$ for $i \in [k]$, and hence $|A_i \,\triangle\, F| \geqslant 1$. This implies, together with (7), the first statement of the claim. Moreover, if $(E, \mathcal{F})$ is an equal-cardinality system, then $A_i \neq F$ implies $|A_i \,\triangle\, F| \geqslant 2$, which leads to the strengthened statement of the claim for equal-cardinality systems. $\qquad\square$

Taking the scalar product of c with both sides of (6), and rearranging terms, we get

$$c(F) - c(Q) = \sum_{i=1}^{k} \lambda_i \cdot (c(F) - c(A_i)) \quad .$$

Using an averaging argument, there exists an index $j \in [k]$ such that

$$\frac{1}{\lambda} \cdot (c(F) - c(Q)) \leqslant c(F) - c(A_j) \quad ,$$

which is equivalent to

$$c(A_j) - c(Q) \leqslant \left(1 - \frac{1}{\lambda} \right) (c(F) - c(Q)) \quad . \tag{8}$$

We will show that $A := A_j$ fulfills the statement of the lemma. First observe that (8) together with the claim implies that A fulfills properties (i) and (i'), respectively. To show (ii) and (ii'), we show that any A_i for $i \in [k]$ fulfills $|Q \,\triangle\, A_i| < |Q \,\triangle\, F|$. This indeed implies both point (ii) and (ii'), because when dealing with

16

equal-cardinality systems, the symmetric difference between any two sets in the system has even cardinality. Hence, it remains to show $|Q \triangle A_i| < |Q \triangle F|$.

We start by observing that equation (6) implies $A_i \subseteq Q \cup F$. Indeed, if there were any $e \in A_i$ with $e \notin Q \cup F$, then this would lead to a strictly positive entry for e on the right-hand side of (6), whereas χ^Q, which appears on the left-hand side of (6), has a 0-entry at e. Analogously, we can derive that $Q \cap F \subseteq A_i$, because if there was $e \in (Q \cap F) \setminus A_i$, then this would imply that the right-hand side of (6) has as its entry at e a value strictly less than 1, contradicting that the left-hand side has a value of 1 at entry e. In summary, we have $A_i \subseteq Q \cup F$ and $Q \cap F \subseteq A_i$. However, among all sets satisfying these properties, the set F is the unique set that maximizes the symmetric difference with Q. Because $A_i \neq F$ we thus have $|Q \triangle A_i| < |Q \triangle F|$, as desired, which finishes the proof of Lemma 15. $\qquad\square$

Lemma 15 is a statement about finding good sets in the 1-neighborhood of any set F. By repeatedly applying the lemma, we obtain the following generalization for q-neighborhoods.

Lemma 16. *Let $(E, \mathcal{F})$ be a set system. Let $c \in \mathbb{R}^E$, and let $F, Q \in \mathcal{F}$ with $F \neq Q$. Then, for any $q \in \{1, \ldots, |Q \triangle F|\}$, there exists a set $A \in \mathcal{F}$ in the q-neighborhood of F on $P_{\mathcal{F}}$ satisfying*

(i) $c(A) - c(Q) \leqslant \left(1 - \frac{q}{|Q \triangle F|}\right) \cdot (c(F) - c(Q))$, and

(ii) $|Q \triangle A| \leqslant |Q \triangle F| - q$.

Moreover, if $(E, \mathcal{F})$ is an equal-cardinality system, then we obtain the following strengthening. For any $q \in \{1, \ldots, \frac{1}{2}|Q \triangle F|\}$, there exists a set $A \in \mathcal{F}$ in the q-neighborhood of F on $P_{\mathcal{F}}$ satisfying

(i') $c(A) - c(Q) \leqslant \left(1 - \frac{2q}{|Q \triangle F|}\right) \cdot (c(F) - c(Q))$, and

(ii') $|Q \triangle A| \leqslant |Q \triangle F| - 2q$.

Proof. We prove the lemma by induction on q. For $q = 1$ the statement holds due to Lemma 15. Now assume $q > 1$, and we show the inductive step for the case where $(E, \mathcal{F})$ is not necessarily an equal-cardinality system. The extension to equal-cardinality systems is analogous. By the inductive hypothesis, there is a set $\overline{A} \in \mathcal{F}$ in the $(q-1)$-neighborhood of F on $P_{\mathcal{F}}$ satisfying

(a) $c(\overline{A}) - c(Q) \leqslant \left(1 - \frac{q-1}{|Q \triangle F|}\right) \cdot (c(F) - c(Q))$, and

(b) $|Q \triangle \overline{A}| \leqslant |Q \triangle F| - (q - 1)$.

Moreover, applying Lemma 15 to $F = \overline{A}$, we obtain that there is a set $A \in \mathcal{F}$ in the 1-neighborhood of $\overline{A}$ in $P_{\mathcal{F}}$—and hence, A is a q-neighbor of F in $P_{\mathcal{F}}$—such that

(c) $c(A) - c(Q) \leqslant \left(1 - \frac{1}{|Q \triangle \overline{A}|}\right) \cdot (c(\overline{A}) - c(Q))$, and

(d) $|Q \triangle A| \leqslant |Q \triangle \overline{A}| - 1$.

The fact that A fulfills point (ii) is now an immediate consequence of (b) and (d). Moreover, we have

$$
\begin{aligned}
c(A) - c(Q) &\leqslant \left(1 - \frac{1}{|Q \triangle \overline{A}|}\right) \cdot \left(1 - \frac{q-1}{|Q \triangle F|}\right) \cdot (c(F) - c(Q)) \\
&\leqslant \left(1 - \frac{1}{|Q \triangle F| - (q-1)}\right) \cdot \left(1 - \frac{q-1}{|Q \triangle F|}\right) \cdot (c(F) - c(Q)) \\
&= \left(1 - \frac{q}{|Q \triangle F|}\right) \cdot (c(F) - c(Q)) \quad,
\end{aligned}
$$

where the first inequality follows from (c) and (a), and the second one from (b). Hence, this shows that A also fulfills (i) and finishes the proof. $\qquad\square$

Our next lemma, Lemma 17, shows that if the value of a set $T \in \mathcal{F}$ is not significantly larger than $c^\top y$, then there is a good solution in its neighborhood. Afterwards, in Lemma 18, we provide a lower bound for the probability of this happening if T has a distribution with marginals given by y, which is the setting of Theorem 14.

Lemma 17. *Let $(E, \mathcal{F})$ be a set system, let $y \in P_{\mathcal{F}}$, $\mu \geqslant 1$, and $\eta = \min\{c(F) \mid F \in \mathcal{F}_y\}$. Then for any $T \in \mathcal{F}_y$ with*

$$(\mu - 1) \cdot (c(T) - \eta) \leqslant \mu \cdot (c^\top y - \eta) \ ,$$

there is a set $U \in \mathcal{F}_y$ in the $\lceil 2\rho(\mathcal{F}_y)/\mu \rceil$-neighborhood of T on $P_{\mathcal{F}_y}$ with $c(U) \leqslant c^\top y$. Moreover, if $(E, \mathcal{F}_y)$ is an equal-cardinality system, then such a set $U \in \mathcal{F}$ even exists in the $\lceil \rho(\mathcal{F}_y)/\mu \rceil$-neighborhood of T on $P_{\mathcal{F}_y}$.

Proof. The statement trivially holds for $\mu = 1$. Hence, assume $\mu > 1$. Let $q := \lceil \rho(\mathcal{F}_y)/\mu \rceil$ if $(E, \mathcal{F})$ is an equal-cardinality system, and $q := \lceil 2\rho(\mathcal{F}_y)/\mu \rceil$ otherwise. Furthermore, we define

$$Q \in \operatorname{argmin}\{c(F) \mid F \in \mathcal{F}_y\} \ ,$$

and hence, $c(Q) = \eta$. By Lemma 16, there is a set $U \in \mathcal{F}_y$ in the q-neighborhood of T on $P_{\mathcal{F}_y}$ satisfying

$$
\begin{aligned}
c(U) - \eta &\leqslant \left(1 - \frac{2\rho(\mathcal{F}_y)}{|Q \triangle T| \cdot \mu}\right) \cdot (c(T) - \eta) \\
&\leqslant \left(1 - \frac{1}{\mu}\right) \cdot (c(T) - \eta) \\
&\leqslant \left(1 - \frac{1}{\mu}\right) \cdot \frac{\mu}{\mu - 1}(c^\top y - \eta) \\
&= c^\top y - \eta \ ,
\end{aligned}
$$

where the second inequality follows from $2\rho(\mathcal{F}_y) \geqslant |Q \triangle T|$, and the third one from the inequality given in the statement of Lemma 17. Hence, U fulfills the properties required by Lemma 17, which finishes the proof. $\square$

Lemma 18. *Let $(E, \mathcal{F})$ be a set system, let $y \in P_{\mathcal{F}}$, $\mu \geqslant 1$, and $\eta = \min\{c(F) \mid F \in \mathcal{F}_y\}$. Moreover, let T be a random set in $\mathcal{F}_y$ drawn from a distribution that satisfies $\Pr[e \in T] = y_e$ for all $e \in E$. Then*

$$\Pr\left[(\mu - 1) \cdot (c(T) - \eta) \leqslant \mu \cdot (c^\top y - \eta)\right] \geqslant \frac{1}{\mu} \ .$$

Proof. First observe that $c(T) - \eta$ is a non-negative random variable with expected value $c^\top y - \eta$, as $\Pr[e \in T] = y_e$ for all $e \in E$. If $c^\top y = \eta$, then the statement holds because $c(T) - \eta$ has expectation zero and is non-negative; thus, it is 0 with probability 1. Assume from now on $c^\top y > \eta$. Then, the lemma is a consequence of Markov's inequality, which implies

$$\Pr\left[c(T) - \eta \geqslant \frac{\mu}{\mu - 1} \cdot (c^\top y - \eta)\right] \leqslant \frac{\mu - 1}{\mu}.$$

Writing $\frac{\mu - 1}{\mu} = 1 - \frac{1}{\mu}$, we see that this implies the statement of the lemma. $\square$

Finally, combining Lemma 17 and Lemma 18, Theorem 14 now readily follows.

Proof of Theorem 14. By choosing $\mu = 2\rho(\mathcal{F})/q$ in both Lemma 17 and Lemma 18 we immediately obtain the first part of Theorem 14. The bound for the case of equal cardinality set systems is obtained by setting $\mu = \rho(\mathcal{F})/q$ in both Lemma 17 and Lemma 18. $\square$

4.2 Further applications of alteration technique

The presented alteration technique is a rather general approach that is not tightly linked to the MCCST setting where we applied it. It may thus be of independent interest. In particular, it can be used to avoid a multiplicative loss in the objective in several contexts where (randomized) rounding approaches are used. We briefly mention one such application. The following packing result was shown in [7].

Theorem 19 (Theorem 6.2 in [7])**.** *Let P be the base polytope of a matroid on ground set N, let $A \in [0,1]^{m \times N}$, and let $b \in \mathbb{R}^N$. Then there is a $(1 + \varepsilon, \mathcal{O}(\log m / \log \log m))$-bicriteria approximation for the problem*

$$\min \left\{ c^\top x \;\middle|\; x \in \{0,1\}^N, \; x \in P, \; Ax \leqslant b \right\} \;,$$

where the first guarantee is w.r.t. the cost of the solution and the second one w.r.t. the overflow on the packing constraints.

The above approximation was obtained by rounding a point $y \in P$ that fulfills $c^\top y \leqslant c(\mathrm{OPT})$ through a negatively correlated rounding procedure. Such a procedure preserves marginals, and hence, falls into the setting of our Theorem 14, which allows for avoiding the loss in the objective, at an additive $+1$ cost in the second objective, which is negligible. Through this alteration, one obtains a unicriteria $(1, \mathcal{O}(\log m / \log \log m))$-approximation.

4.3 Alternative approach to avoid $1 + \varepsilon$ loss via techniques of Linhares and Swamy

We briefly want to highlight that a recently introduced approach of Linhares and Swamy [20] also allows for avoiding a $(1 + \varepsilon)$-factor loss in the objective. More precisely, they introduced a Lagrangian relaxation based approach to reduce certain bicriteria weighted packing problems to bicriteria unweighted packing problems. Within this framework, they also show how it can be modified to avoid losses in the objective value under some conditions. For this they need an LP-based rounding procedure with certain properties.

In the following, we focus on the specific problem of MCCST to expand further on this approach and how it can be made to work in this context. For simplicity, consider an MCCST problem with only upper bounds on the chain constraints, which falls within the setting of packing constraints considered in [20]. Let $y \in Q$ be a fractional point as computed in the first step of Algorithm 1. The point y can be interpreted as an optimal LP solution to a linear program on the minimal face of the spanning tree polytope on which y lies, together with upper bounds on the chain constraints of large y-value. This allows for interpreting y as an LP solution as required by the framework of Linhares and Swamy. Additionally, the framework needs a rounding procedure that both (i) rounds y to a spanning tree T on the same minimal face of the spanning tree polytope on which y lies, and (ii) the spanning tree T needs to satisfy that $|T \cap \delta(S_i)|$ is within a $(1 \pm \epsilon)$-factor of b_i for each chain constraint corresponding to a set S_i for which y is tight, i.e., $y(\delta(S_i)) = b_i$. Notice that it is important that the rounding does not just return a tree almost fulfilling the chain constraints, but we also need that y-tight chain constraints remain nearly-tight after rounding. Our alteration step does not require such a property, but has other requirements. Hence, the two techniques are not strictly comparable.

The negatively correlated rounding procedure that we employ fulfills both requirements stated above. It always rounds to a spanning tree on the same minimal face, because it is marginal-preserving. Moreover, equation (2) shows that the load on chain constraints does not change much.

Finally, we want to mention that the framework in [20] can also be adjusted to deal with lower bounds in our context of MCCST.

5 Extension to MLCST

The key ingredient of the approximation algorithm for MCCST presented in the previous sections is obtaining a τ-integral point y that is feasible for the linear relaxation of the problem. Ideally, we would like to find such a point in the more general case of MLCST as well, and then apply Theorems 5 and 8 for rounding and local corrections, as before.

Unfortunately, analyzing the natural generalization of our dynamic programming approach, where we determine partial solutions $y_{(S,F,\mathcal{C})}$ for all connectivity triples $(S, F, \mathcal{C})$ with $S \in \mathcal{L}$ by continuously extending previously obtained solutions, comes with obstacles even if the laminar family $\mathcal{L}$ has constant width. To highlight some aspects thereof, consider the instance given in Figure 3, where the laminar family $\mathcal{L}$ consists of precisely two sets S_1 and S_2, and edge costs c are such that edges in $E[S_1] \cup E[S_2]$ have cost 1, and all other edges have cost 0.
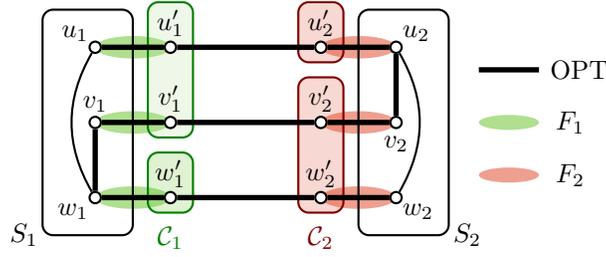
Figure 3: Connectivity triples $(S_1, F_1, \mathcal{C}_1)$ and $(S_2, F_2, \mathcal{C}_2)$ induced by an optimal solution OPT.

It is easy to see that every optimal solution has cost 2, as for example the one indicated in bold. Note that in our dynamic program, we would first construct solutions $y_{(S_i,F,\mathcal{C})}$ compatible with the pattern $(S_i, F, \mathcal{S})$ for all such connectivity patterns, and then try to extend every combination of solutions on S_1 and S_2 to a global solution. Typically, such dynamic programming approaches are analyzed by backtracing an optimal solution. In our example, consider the optimal solution given in Figure 3. This solution induces the connectivity triples $(S_1, F_1, \mathcal{C}_1)$ and $(S_2, F_2, \mathcal{C}_2)$ on S_1 and S_2, respectively, where $F_i = \{\{u_i, u_i'\}, \{v_i, v_i'\}, \{w_i, w_i'\}\}$ for $i \in \{1, 2\}$, $\mathcal{C}_1 = \{\{u_1', v_1'\}, \{w_1'\}\}$ and $\mathcal{C}_2 = \{\{u_2'\}, \{v_2', w_2'\}\}$, as indicated. Ideally, we would like that the cheapest common extension of $y_{(S_1,F_1,\mathcal{C}_1)}$ and $y_{(S_2,F_2,\mathcal{C}_2)}$ has cost at most the cost of an optimal solution. However, note that we could potentially have

$$y_{(S_1,F_1,\mathcal{C}_1)} = \chi^{\{\{u_1,w_1\}\}} \quad \text{and} \quad y_{(S_2,F_2,\mathcal{C}_2)} = \chi^{\{\{u_2,w_2\}\}},$$

in which case we can easily see that there does not even exist a feasible solution that restricts to $y_{(S_1,F_1,\mathcal{C}_1)}$ and $y_{(S_2,F_2,\mathcal{C}_2)}$ on S_1 and S_2, respectively.

Note that in the case of MCCST, where we only had to consider extensions from a single partial solution on a smaller set, the analysis outlined above was enough to obtain our result (see the proof of Lemma 12). In particular, it was enough to know that the dynamic program considered building a solution along the small cuts and connectivity triples induced by an optimal solution (even though these are not known upfront). For MLCST, we deviate from this typical analysis and exploit that the DP considers all potential connectivity triples on the small cuts induced by an optimal solution. Let us illustrate this in the above example. We claim that there exist connectivity patterns $\mathcal{C}_1'$ and $\mathcal{C}_2'$ (potentially different from the patterns $\mathcal{C}_1$ and $\mathcal{C}_2$ induced by the optimal solution) such that the best common extension of $y_{(S_1,F_1,\mathcal{C}_1')}$ and $y_{(S_2,F_2,\mathcal{C}_2')}$ has cost at most 2. To see this, we proceed iteratively, starting with $\mathcal{C}_1' = \mathcal{C}_1$. Note that if on $E[S_1]$, we replace the edges of the optimal solution by those of $y_{(S_1,F_1,\mathcal{C}_1)}$, the new point is a feasible solution and, by definition of $y_{(S_1,F_1,\mathcal{C}_1)}$, the total cost does not increase. Observe that the new set of edges induces a different connectivity pattern on S_2 than OPT did, and let this pattern be $\mathcal{C}_2'$ (see Figure 4). Now replacing the optimal solution on $E[S_2]$ by

$y_{(S_2,F_2,\mathcal{C}'_2)}$, which is $\chi^{\{\{u_2,v_2\}\}}$ in our example, we again see that feasibility is guaranteed, and the total cost does again not increase. To finish the argument, note that we just constructed a common extension of the two partial solutions $y_{(S_1,F_1,\mathcal{C}'_1)}$ and $y_{(S_2,F_2,\mathcal{C}'_2)}$ of cost no more than the cost of the optimal solution—thus, the best extension will be of cost at most OPT, proving the desired guarantee.
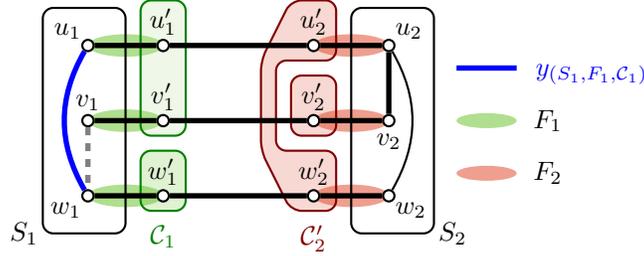
Figure 4: Patching $y_{(S_1,F_1,\mathcal{C}_1)}$ induces $\mathcal{C}'_2$.

The above idea of iteratively defining suitable connectivity patterns can be generalized to an arbitrary number of sibling sets $S_1,\ldots,S_w$, and is crucial in the analysis of the propagation step of our dynamic program. For now, we want to highlight that replacing parts of an optimal solution by a previously obtained partial solution and then inducing new connectivity patterns on other parts requires the partial solutions to be integral, as there is no notion of induced connectivity patterns for fractional solutions. In our extension steps, however, we find common extensions of partial solutions through a linear program similar to (exLP), which will in general not be integral. For this reason, we apply the rounding and local correction methods presented in Theorems 5 and 8 after every single extension step, giving integral solutions at every stage and thus allowing for inducing connectivity patterns iteratively as defined above.

The example from Figure 3 discussed earlier might seem contrived due to the fact that the highlighted problems could be avoided by breaking ties in the right way, for example by choosing different optimal partial solutions $y_{(S_1,F_1,\mathcal{C}_1)}$ and $y_{(S_2,F_2,\mathcal{C}_2)}$. Even though this is the case here, there exist more complex examples without any tie-breaking options that exhibit the same issue. One such example is presented in Appendix B.

It is important to also recall that Theorems 5 and 8 provide guarantees on constraint violation and cost of the integral solution obtained through rounding only with certain probabilities. For a more concise analysis, our algorithm will at each step apply the rounding and local correction operations repeatedly until—in expected polynomial running time—an integral solution with the desired properties is found. More formally, we obtain a Las Vegas algorithm with the following guarantees.

Theorem 20. *For every $\varepsilon > 0$, there is a $(1, 1 + \varepsilon)$-approximation algorithm for MLCST with expected running time $|V|^{\mathcal{O}(k \log |V|/\varepsilon^2)}$, where k is the width of the laminar family $\mathcal{L}$.*

Note that any Las Vegas algorithm can easily be transformed into a randomized approximation algorithm, i.e, with deterministic polynomial running time and where the returned solution has the desired properties with high probability: By Markov's inequality, the probability that the running time of a single run of the algorithm guaranteed by Theorem 20 is less that twice the expected running time is at least $1/2$. Consequently, among $\log_2 |V|$ many independent runs, the probability that at least one run succeeds is at least $1 - 1/|V|$. Thus, Theorem 2 stated in the introduction is implied by Theorem 20.

In the following two sections, we present the modifications of our dynamic programming approach for the laminar case in detail, and we expand on the ideas highlighted above, leading to a proof of Theorem 20.

5.1 Dynamic programming in the laminar case

For a formal description of the dynamic programming algorithm for MLCST, we stick to the notation defined for MCCST, now of course considering the laminar family $\mathcal{L}$ instead of the chain $S_1 \subsetneq \ldots \subsetneq S_k$. We denote

by $\mathcal{K}$ the set of all connectivity triples $(S, F, \mathcal{C})$ where $S \in \mathcal{L} \cup \{\emptyset, V\}$, with the crossing edges $F \subseteq \delta(S)$ satisfying $a_S \leqslant |F| \leqslant \min\{\tau, b_S\}$ for $S \in \mathcal{L}$, where the parameter τ is the maximal number of edges that the dynamic program "guesses" in small cuts. For MLCST, we will again choose $\tau = \mathcal{O}(\log |V|/\varepsilon^2)$.

For every $(S, F, \mathcal{C}) \in \mathcal{K}$, we use our dynamic program to determine a partial solution $T_{(S,F,\mathcal{C})} \subseteq E[S]$ with the following properties. For $S \in \mathcal{L}$, we denote $\mathcal{L}_S := \{S' \in \mathcal{L} \mid S' \subsetneq S\}$.

Property 21.

(i) $T_{(S,F,\mathcal{C})}$ is a spanning tree of $G(S, F, \mathcal{C})$.

(ii) $(1 - \varepsilon) \cdot a_{S'} \leqslant |(T_{(S,F,\mathcal{C})} \cup F) \cap \delta(S')| \leqslant (1 + \varepsilon) \cdot b_{S'}$ for all $S' \in \mathcal{L}_S$.

(iii) $c(T_{(S,F,\mathcal{C})})$ is at most the cost of a cheapest edge set $U \subseteq E[S]$ that forms a spanning tree of $G(S, F, \mathcal{C})$ and satisfies $a_{S'} \leqslant |(U \cup F) \cap \delta(S')| \leqslant b_{S'}$ for all $S' \in \mathcal{L}_S$.

It is clear that if in expected running time $|V|^{\mathcal{O}(k \log |V|/\varepsilon^2)}$, we can obtain trees $T_{(S,F,\mathcal{C})}$ with the above properties, then Theorem 20 follows because the tree $T := T_{(V,\emptyset,\emptyset)}$ has precisely the desired properties: (i) and (ii) state that T is a spanning tree of G violating the cut constraints only by a factor of $(1 \pm \varepsilon)$, and by (iii), $c(T)$ is at most the cost of an optimal solution.

To compute all $T_{(S,F,\mathcal{C})}$, we initialize $T_{(\emptyset,\emptyset,\{\emptyset\})} = \emptyset$, and then propagate to $T_{(S,F,\mathcal{C})}$ for all $(S, F, \mathcal{C}) \in \mathcal{K}$ in an order such that for all $S, S' \in \mathcal{L}$, if $S' \subsetneq S$, then $T_{(S',F,\mathcal{C})}$ is computed before $T_{(S,F,\mathcal{C})}$. A new tree $T_{(S,F,\mathcal{C})}$ is obtained in two steps: First, for all choices of connectivity triples $(S_1, F_1, \mathcal{C}_1), \ldots, (S_w, F_w, \mathcal{C}_w)$ such that $S_1, \ldots, S_w \in \mathcal{L}_S$ and $S_1, \ldots, S_w$ have pairwise empty intersections, we extend the partial solutions $T_{(S_i,F_i,\mathcal{C}_i)}$ for $i \in [w]$ to a solution T on $(S, F, \mathcal{C})$. In a second step, we find the best solution among all extensions obtained this way, and keep it as $T_{(S,F,\mathcal{C})}$. The full propagation procedure is summarized in Algorithm 4, and details of a single extension step are described in Algorithm 3.

Algorithm 3: Extending partial solutions $T_{(S_1,F_1,\mathcal{C}_1)}, \ldots, T_{(S_w,F_w,\mathcal{C}_w)}$ to $(S, F, \mathcal{C})$.

---

1. Let $\tau := \lfloor 96 \log(2|V|)/\varepsilon^2 \rfloor$, and let y be a minimizer of

$$\min \quad c^\top y \tag{lamExLP}$$
$$y \in P_{\mathrm{ST}}(S_i, F, \mathcal{C})$$
$$\max\{\tau + 1, a_{S'}\} \leqslant y(\delta(S')) + |F \cap \delta(S')| \leqslant b_{S'} \quad \forall S' \in \mathcal{L}: \exists i \in [w] \text{ with } S_i \subsetneq S' \subsetneq S$$
$$y(e) = \chi^{T_{(S_i,F_i,\mathcal{C}_i)}}(e) \quad \forall e \in E[S_i], \ \forall i \in [w]$$
$$y(e) = \chi^{F_i}(e) \quad \forall e \in \delta(S_i) \setminus F, \ \forall i \in [w] \ .$$

2. Randomly round y with a rounding procedure as guaranteed by Theorem 5 to obtain a spanning tree T_0 of $G(S, F, \mathcal{C})$.

3. Find a minimum cost spanning tree T among all spanning trees of $G(S, F, \mathcal{C})$ with $|T \triangle T_0| \leqslant 2$ and such that $y(e) \in (0, 1)$ for all $e \in T \triangle T_0$.

4. If T satisfies
$$(1 - \varepsilon) \cdot a_{S'} \leqslant |(T_{(S,F,\mathcal{C})} \cup F) \cap \delta(S')| \leqslant (1 + \varepsilon) \cdot b_{S'}$$

for all $S' \in \mathcal{L}_S$, and $c(T) \leqslant c^\top y$, output T. Else, repeat from step 2.

Let us expand on the nature of the extension step given in Algorithm 3. The purpose of the linear program (lamExLP) is to find a common extension y of the trees $T_{(S_i,F_i,\mathcal{C}_i)}$ for $i \in [w]$ that uses precisely the edges F_i in the cuts $\delta(S_i)$, and is left-compatible with $(S, F, \mathcal{C})$. Note that the last condition appears in (lamExLP) as the constraint $y \in P_{\mathrm{ST}}(S, F, \mathcal{C})$. Additionally, we require that the partial solution y together with the edges in F satisfy the cut constraints on cuts $S' \in \mathcal{L}$ with $S_i \subsetneq S' \subsetneq S$, with a load of at least

$\tau + 1$ on all those cuts. Recall that the latter comes from the idea of finding τ-integral fractional solutions (which is what we need to control cut sizes in the rounding procedure) and letting the dynamic program try all combinations of maximal small cuts $S_1, \ldots, S_w \subsetneq S$.

Once a fractional extension y is found, we use a rounding procedure as guaranteed by Theorem 5 to round it to an integral solution, namely a spanning tree T_0. Note that by definition, y coincides with $T_{(S_i, F_i, C_i)}$ on $E[S_i]$ for all $i \in [w]$. As the rounding scheme is marginal-preserving (property (i) in Theorem 5), it follows that T_0 will coincide with these partial solutions as well, thus inheriting their properties. On $E[S \setminus \bigcup_{i=1}^{w} S_i]$, property (ii) will make sure that all cut constraints are satisfied up to small multiplicative errors. In step 3, we exploit the exchange steps described in Theorem 8 to regain potential loss in the objective that may have occurred in step 2 compared to $c^\top y$. Both step 2 and step 3 can fail with certain probabilities, hence we repeatedly apply them until an extension with the properties listed in step 4 is found.

We remark that the linear program (lamExLP) might be infeasible for several reasons (impossibility of completing the edge sets $T_{(S_i, F_i, C_i)}$ to a point in $P_{\mathrm{ST}}(S, F, C)$, infeasibility of the lower bound constraints on cuts, inconsistencies among the edge sets F_i, loops generated by edges in F_i, etc.). In such a case, we interpret the cost of a common integral extension T to be ∞, which avoids using such extensions later on.

Algorithm 4: Propagation to $T_{(S,F,C)}$ from all partial solutions $T_{(S',F',C')}$ with $S' \subsetneq S$.

---

1. For every choice of connectivity triples $(S_i, F_i, C_i) \in \mathcal{K}'$ for $i \in [w]$ where $S_1, \ldots, S_w \in \mathcal{L}$ are strict subsets of S with pairwise empty intersections, apply Algorithm 3 to extend the trees $T_{(S_1, F_1, C_1)}, \ldots, T_{(S_w, F_w, C_w)}$ to (S, F, C). Let $\mathcal{T}$ be the set of all trees obtained this way.
2. Return $T_{(S,F,C)} \in \mathrm{argmin}_{T \in \mathcal{T}} c(T)$.

Algorithm 4 considers all potential candidates for $T_{(S,F,C)}$ that were obtained through extension steps, and returns the one of minimum cost.

5.2 Analyzing the DP

We first show that the trees $T_{(S,F,C)}$ computed by Algorithm 3 satisfy Property 21. Formally, we prove this statement by induction. Obviously, the point $T_{(\emptyset,\emptyset,\{\emptyset\})}$ has all desired properties. The induction step to complete the proof is captured by the following lemma.

Lemma 22. *Let $(S, F, C) \in \mathcal{K}$ with $S \neq \emptyset$. Assume that for all $(S', F', C') \in \mathcal{K}$ with $S' \subsetneq S$, we are given $T_{(S',F',C')}$ satisfying Property 21, and let $T_{(S,F,C)}$ be obtained from Algorithm 4. Then $T_{(S,F,C)}$ satisfies Property 21, as well.*

Proof. Let $\mathcal{T}$ be defined as in Algorithm 4, namely the set of all $T \subseteq E[S]$ that were obtained trough Algorithm 3. We already remarked earlier that any such T is a spanning tree of the corresponding graph $G(S, F, C)$, hence satisfying point (i) in Property 21. For point (ii), we prove that every $T \in \mathcal{T}$ satisfies

$$(1 - \varepsilon) \cdot a_{S'} \leqslant |(T \cup F) \cap \delta(S')| \leqslant (1 + \varepsilon) \cdot b_{S'} \tag{9}$$

for all $S' \in \mathcal{L}_S$. Indeed, for cuts S' with $S' \subsetneq S_i$ for some $i \in [w]$, we have $(T \cup F) \cap \delta(S') = (T_{(S_i, F_i, C_i)} \cup F_i) \cap \delta(S_i)$, hence (9) follows from the assumption that $T_{(S_i, F_i, C_i)}$ has Property 21. If $S' = S_i$ for some $i \in [w]$, then $(T \cup F) \cap \delta(S') = F_i$, and we have $a_{S'} \leqslant |F_i| \leqslant b_{S'}$ by definition of the connectivity pattern (S_i, F_i, C_i). Finally, if $S_i \subsetneq S' \subsetneq S$, then (9) is guaranteed by step 4 in Algorithm 3.

To see that point (iii) of Property 21 holds, fix an edge set $U \subseteq E[S]$ that forms a spanning tree of $G(S, F, C)$ and satisfies $a_{S'} \leqslant |(U \cup F) \cap \delta(S')| \leqslant b_{S'}$ for all $S' \in \mathcal{L}$ with $S' \subsetneq S$. We have to show that $c(T_{(S,F,C)}) \leqslant c(U)$. As a first step, consider any $T \in \mathcal{T}$ and let y be the solution of (lamExLP) that was used

to obtain T. By step 4 in Algorithm 3, we have $c(T) \leqslant c^\top y$. It is thus enough to see that one of the linear programs (lamExLP) considered while propagating to $(S, F, \mathcal{C})$ has a solution y with $c(U) \geqslant c^\top y$.

To this end, let $R \subseteq \binom{V \setminus S}{2}$ be any set of edges that is right-compatible with $(S, F, \mathcal{C})$. Then, $T :=$ $U \cup F \cup R$ is in the spanning tree polytope of $(V, E \cup R)$. Let $S_1, \dots, S_w \subsetneq S$ be the maximal χ^T-small cuts in $\mathcal{L}_S$, and let $F_i = T \cap \delta(S_i)$. We define connectivity patterns $\mathcal{C}_1, \dots, \mathcal{C}_w$ such that $(S_i, F_i, \mathcal{C}_i) \in \mathcal{K}$ iteratively as follows, where $T_0 := T$ and $i \in [w]$:

$$\text{Let } \mathcal{C}_i \text{ s.t. } T_{i-1} \text{ is compatible with } (S_i, F_i, \mathcal{C}_i), \text{ and let } T_i := \left(T_{i-1} \setminus E[S_i]\right) \cup T_{(S_i, F_i, \mathcal{C}_i)}. \tag{10}$$

First of all, observe that all edge sets T_i are indeed spanning trees, making the above operation well-defined. To see this, we proceed inductively and assume that T_{i-1} is a spanning tree. Compatibility of T_{i-1} with $(S_i, F_i, \mathcal{C}_i)$ implies that $T_{i-1} \cap E[V \setminus S_i]$ is right-compatible with $(S_i, F_i, \mathcal{C}_i)$, while $T_{(S_i, F_i, \mathcal{C}_i)}$ is left-compatible with $(S_i, F_i, \mathcal{C}_i)$, by assumption. Thus, $(T_{i-1} \cap E[V \setminus S_i]) \cup F_i \cup T_{(S_i, F_i, \mathcal{C}_i)} = (T_{i-1} \setminus E[S_i]) \cup T_{(S_i, F_i, \mathcal{C}_i)} = T_i$ is indeed a spanning tree. We claim that the construction in (10) leads to a tree T_w with the properties

(a) $c(T) \geqslant c(T_w)$, and

(b) $T_w \cap E[S]$ is feasible for (lamExLP) when extending from $T_{(S_1, F_1, \mathcal{C}_1)}, \dots, T_{(S_w, F_w, \mathcal{C}_w)}$ to $(S, F, \mathcal{C})$.

These two properties are enough to conclude. As T and T_w are identical outside of $E[S]$, (a) implies that

$$c(U) = c(T \cap E[S]) \geqslant c(T_w \cap E[S]) \ .$$

Moreover, (b) implies that $c(T_w \cap E[S]) \geqslant c^\top y$, which together with the previous inequality gives the desired $c(U) \geqslant c^\top y$.

To see property (a), we show that for all $i \in [w]$, we have $c(T_{i-1}) \geqslant c(T_i)$. By definition of T_i, the latter is equivalent to $c(T_{i-1} \cap E[S_i]) \geqslant c(T_i \cap E[S_i])$. But by construction, $T_{i-1} \cap E[S_i] = U \cap E[S_i]$, and $T_i \cap E[S_i] = T_{(S_i, F_i, \mathcal{C}_i)}$. Note that $U \cap E[S_i]$ is an integral solution of the subproblem on $G(S_i, F_i, \mathcal{C}_i)$, and hence Property 21 (iii) for $T_{(S_i, F_i, \mathcal{C}_i)}$ implies $c(U \cap E[S_i]) \geqslant c(T_{(S_i, F_i, \mathcal{C}_i)})$, which is precisely what we need. Consequently, we have $c(T_{i-1}) \geqslant c(T_i)$ for all $i \in [w]$, and hence

$$c(T) = c(T_0) \geqslant c(T_1) \geqslant \dots \geqslant c(T_w) \ ,$$

as desired. For property (b), we check that the constraints in (lamExLP) hold for $T_w \cap E[S]$, i.e., for all $S' \in \mathcal{L}$ such that there is an $i \in [w]$ with $S_i \subsetneq S' \subsetneq S$, we prove

$$\max\{\tau + 1, a_{S'}\} \leqslant |(T_w \cap E[S]) \cap \delta(S')| + |F \cap \delta(S')| \leqslant b_{S'} \ . \tag{11}$$

We have $|(T_w \cap E[S]) \cap \delta(S')| + |F \cap \delta(S')| = |T_w \cap \delta(S')| = |(U \cup F) \cap \delta(S')|$, where the last equality follows from the construction of T_w. Thus, the lower bound $a_{S'}$ and the upper bound $b_{S'}$ are implied by the assumption on U. Moreover, the lower bound $\tau + 1$ follows from the definition of $S_1, \dots, S_w$ as the maximal $\chi^{U \cup F}$-small cuts, implying (11). Finally, by construction, T_w equals $T_{(S_i, F_i, \mathcal{C}_i)}$ and F_i on $E[S_i]$ and $\delta(S_i)$, respectively, for all $i \in [w]$. Consequently, $T_w \cap E[S]$ satisfies all constraints of (lamExLP). This finishes the proof of Lemma 22. □

It remains to analyze the expected running time of an extension step as described in Algorithm 3.

Lemma 23. *Algorithm 3 has expected running time* $|V|^{\mathcal{O}(1)}$.

Proof. First of all, note that every single step of Algorithm 3 can be implemented in running time $|V|^{\mathcal{O}(1)}$. In particular, linear programs of the type (lamExLP) can be solved in strongly polynomial time by using a compact extended formulation for the spanning tree polytope with small coefficients in the constraint matrix

(one can, for example, use the one by Martin [21], which has coefficients that are bounded by 1 in absolute value), and then applying the framework of Tardos [29]. Consequently, the above lemma is reduced to proving a bound on the expected number of iterations that are needed to achieve the properties required in step 4.

Replicating the analysis in the proof of Theorem 1, we see that Theorems 5 and 8 imply that the tree T obtained in steps 2 and 3 of Algorithm 3 has the desired properties with probability at least $1/2|V|$. As all iterations are independent, the probability that we succeed precisely at iteration $j \in \mathbb{Z}_{>0}$, i.e., in time $j \cdot |V|^{\mathcal{O}(1)}$, is $(1 - 1/2|V|)^{j-1} \cdot 1/2|V|$. Consequently, the expected running time is

$$\sum_{j \geqslant 1} j \cdot |V|^{\mathcal{O}(1)} \cdot \left(1 - \frac{1}{2|V|}\right)^{j-1} \cdot \frac{1}{2|V|} = |V|^{\mathcal{O}(1)} \ ,$$

where we use that $\sum_{j \geqslant 1} j(1-x)^{j-1} = \frac{1}{x^2}$ for $x \in (0,1)$. $\qquad\square$

Together with the bound $|\mathcal{K}| = |V|^{\mathcal{O}(\tau)}$ from Proposition 11, we are finally ready to prove Theorem 20.

Proof of Theorem 20. We run a dynamic program that calculates trees $T_{(S,F,\mathcal{C})}$ for all $(S,F,\mathcal{C}) \in \mathcal{K}$ starting from the initialization $T_{(\emptyset,\emptyset,\{\emptyset\})} = \emptyset$, and using Algorithm 4 for propagation. Note that $T_{(\emptyset,\emptyset,\{\emptyset\})}$ satisfies Property 21, and hence by an inductive application of Lemma 22, all trees $T_{(S,F,\mathcal{C})}$ satisfy Property 21. In particular, $T_{(V,\emptyset,\{\emptyset\})}$ is thus a tree satisfying the guarantees of Theorem 20.

The running time is determined by the number of calls to Algorithm 3. For every triple $(S,F,\mathcal{C}) \in \mathcal{K}$, when calculating $T_{(S,F,\mathcal{C})}$, there is one call to Algorithm 3 for every possible choice of triples $(S_1, F_1, \mathcal{C}_1), \ldots, (S_w, F_w, \mathcal{C}_w) \in \mathcal{K}$ with $S_1, \ldots, S_w \in \mathcal{L}_S$ having pairwise empty intersections. Note that $\mathcal{L}$ has width k, so there are at most k sets with pairwise empty intersection, i.e., $w \leqslant k$. This implies that the number of calls to Algorithm 3 is bounded from above by $|\mathcal{K}|^{k+1}$. Consequently, the bound on the expected running time of $|V|^{\mathcal{O}(k\tau)}$ follows from combining Proposition 11 and Lemma 23. $\qquad\square$

6 Implications in Path TSP and beyond

In this section, we present an application of our new dynamic programming technique to Path TSP and one natural generalization of it, namely the problem of finding connected T-joins. We remark that in order to keep notation unambiguous, we reserve the variable T for trees and use Q instead in Q-joins throughout the rest of this paper.

Observe that the solution structure in the connected Q-join problem is more general than for Path TSP, as by definition, Q-joins are edge multisets, while a path can contain every edge at most once. However, it turns out that the connected Q-join problem always has optimal solutions with simpler structure, as the following theorem shows.

Theorem 24 (Cheriyan, Friggstad, Gao [9])**.** *Let $G = (V, E)$ be a complete graph with metric edge lengths $\ell \colon E \to \mathbb{R}_{\geqslant 0}$, and let $Q \subseteq V$ be non-empty and of even cardinality. Given a connected Q-join J, a spanning tree T of G with $\ell(T) \leqslant \ell(J)$ that is a connected Q-join can be found efficiently.*

The proof of Theorem 24 exploits the assumption that the instance is metric, and shows that whenever a Q-join has parallel edges or cycles, shortcutting allows for a reduction. In particular, Theorem 24 shows that indeed, Path TSP is a special case of the connected Q-join problem, as for $Q = \{s, t\}$ with $s \neq t$, a spanning tree that is a connected Q-join is a Hamiltonian s-t path. Moreover, by Theorem 24, we also see that it is enough for an approximate solution to compare well to optimal spanning trees, which is a crucial observation for simplifying our analysis.

With the above said, we give a short recap of the approach by Christofides' for TSP and Path TSP [10] in the context of the more general shortest connected Q-join problem. More precisely, we walk through a

polyhedral analysis of Christofides' Algorithm due to Wolsey [32]. These ideas form the basis of essentially all improvements in approximation algorithms for Path TSP over the last few years [1, 15, 26, 30, 31], culminating in a recent $3/2$-approximation algorithm for Path TSP by Zenklusen [34].

6.1 Christofides' algorithm and Wolsey's analysis

Christofides' algorithm builds on the observation that a solution has to satisfy two properties, namely connectivity and correct degree parities. Connectivity can be guaranteed by starting with a spanning tree T. The degree parities of T are wrong precisely at the vertices in $Q_T := \mathrm{odd}(T) \triangle Q$, which can be corrected by adding a Q_T-join J to T.[10] The multiset obtained by combining T and J can be shortcut to a solution of the problem, and approximation guarantees follow by choosing T and J such that $\ell(T)$ and $\ell(J)$ can be bounded in terms of $\ell(\mathrm{OPT})$, where OPT denotes an optimal solution of the problem.

While it is easy to observe that for a shortest spanning tree T, we have $\ell(T) \leqslant \ell(\mathrm{OPT})$, bounds on $\ell(J)$ for a shortest Q_T-join J can for example be obtained by exploiting polyhedral descriptions of Q-joins. In particular, the dominant of the Q-join polytope[11] is given by

$$P_{Q\text{-join}}^{\uparrow} := \left\{ x \in \mathbb{R}_{\geqslant 0}^{E} \,\middle|\, x(\delta(C)) \geqslant 1 \; \forall\, Q\text{-cuts } C \subseteq V \right\} \;,$$

where a Q-cut C is a subset of V with $|C \cap Q|$ odd (see [24, Section 29]). By integrality of this polytope, in order to prove a bound of the form $\ell(J) \leqslant \gamma \cdot \ell(\mathrm{OPT})$ for a shortest Q_T-join J, it is sufficient to find a point $z \in P_{Q_T\text{-join}}^{\uparrow}$ with $\ell^{\top} z \leqslant \gamma \cdot \ell(\mathrm{OPT})$, and a $(1 + \gamma)$-approximation follows.

In many approaches, an important role for finding a suitable point z is taken by a linear relaxation of the problem. For the connected Q-join problem, we use the formulation

$$
\begin{aligned}
\min \quad & \ell^{\top} x \\
& x(\delta(C)) \geqslant 2 \quad \forall C \subsetneq V,\ C \neq \emptyset,\ |C \cap Q| \text{ even} \\
& x(\delta(C)) \geqslant 1 \quad \forall C \subseteq V,\ |C \cap Q| \text{ odd} \\
& x \in \mathbb{R}_{\geqslant 0}^{E} \;,
\end{aligned}
\tag{$\mathrm{LP_{HK}}$}
$$

which is an adaptation of the well-known Held-Karp relaxation for TSP.

If x^* is an optimal solution of ($\mathrm{LP_{HK}}$), the point $z = x^*/2$ is a good candidate for a feasible point of $P_{Q_T\text{-join}}^{\uparrow}$. More precisely, observe that the constraint $z(\delta(C)) \geqslant 1$ is violated precisely for cuts C with $x^*(\delta(C)) < 2$. As $\delta(C) = \delta(V \setminus C)$, we can fix a vertex $q \in Q$, and it is enough to consider the cuts in the family

$$\mathcal{N} := \{ C \subseteq V \mid q \notin C,\ x^*(\delta(C)) < 2 \} \;,$$

the so-called *narrow* cuts of x^*. By the first constraint in ($\mathrm{LP_{HK}}$), only cuts C with $|C \cap Q|$ odd can be narrow. Moreover, note that the constraints $z(\delta(C)) \geqslant 1$ only appear in the description of $P_{Q_T\text{-join}}^{\uparrow}$ if C is a Q_T-cut, but this is not necessarily the case for all $C \in \mathcal{N}$. If indeed, none of the narrow cuts are Q_T-cuts, we conclude that $z = x^*/2$ is feasible for $P_{Q_T\text{-join}}^{\uparrow}$. Using that $\ell^{\top} x^* \leqslant \ell(\mathrm{OPT})$, we get

$$\ell(T) + \ell(J) \leqslant 3/2 \cdot \ell(\mathrm{OPT}) \;,$$

and hence a $3/2$-approximation. In particular, if we could obtain an optimal solution x^* of ($\mathrm{LP_{HK}}$) and a tree T^* with $\ell(T^*) \leqslant \ell(\mathrm{OPT})$ that has an odd number of edges in every narrow cut of x^*, we could achieve the above result. To see this, consider a narrow cut C, and observe that

$$\sum_{v \in C} \deg_{T^*}(v) = 2 \cdot |T^* \cap E[C]| + |T^* \cap \delta(C)| \;.$$

---

[10] By $\mathrm{odd}(T)$, we denote the set of odd-degree vertices in T.

[11] The dominant of the Q-join polytope is the set of all points $x \in \mathbb{R}^{E}$ such that there is a convex combination $y = \sum_{i=1}^{k} \lambda_i \chi^{J_i}$ of characteristic vectors $\chi^{J_i} \in \{0, 1\}^{E}$ of Q-joins J_i with $\lambda_i > 0$ for $i \in [k]$ such that $y \leqslant x$.

The assumption that $|T^* \cap \delta(C)|$ is odd implies that T^* has an odd number of odd-degree vertices in C. As $|C \cap Q|$ is odd, we conclude that $|C \cap Q_{T^*}| = |C \cap (\mathrm{odd}(T^*) \triangle Q)|$ is even, i.e., C is indeed not a Q_{T^*}-cut.

Ideally, we would thus like to find a spanning tree T^* that has an odd number of edges in each narrow cut $C \in \mathcal{N}$. Cheriyan, Friggstad, and Gao [9] showed that the family $\mathcal{N}$ is in fact a laminar family. We additionally observe that the width of this family is at most $|Q| - 1$: Every cut $C \in \mathcal{N}$ has odd intersection with Q, i.e., it contains at least one element of $Q \setminus \{q\}$, so there can be at most $|Q| - 1$ many mutually disjoint sets in $\mathcal{N}$. However, even for a chain, one can see that it is NP-hard to decide whether there is a spanning tree that is odd in each cut.[12] As an alternative, we can also use a tree T with slightly weaker properties, and instead modify the vector z to obtain a feasible point for $P^{\uparrow}_{Q_T\text{-join}}$. In particular, observe the following.

Observation 25. *Assume that we are given an optimal solution x^* of $(\mathrm{LP_{HK}})$, a tree T and a point $y \in \mathbb{R}^E_{\geqslant 0}$ with the following properties.*

    (i) For all narrow cuts C of x^*, either $|T \cap \delta(C)|$ is odd, or $y(\delta(C)) \geqslant 1/\varepsilon$.

    (ii) $\ell^\top y \leqslant \ell(\mathrm{OPT})$, and $\ell(T) \leqslant \ell(\mathrm{OPT})$.

Then, shortcutting the combination of T and a shortest Q_T-join J gives a $(1.5 + \varepsilon)$-approximate solution for the connected Q-join problem.

Proof. We claim that $z := \frac{1}{2}(x^* + \varepsilon y) \in \mathbb{R}^E_{\geqslant 0}$ satisfies $z \in P^{\uparrow}_{T_Q\text{-join}}$. From the above discussion and the bounds in (ii), this immediately implies the observation. The constraints of the form $z(\delta(C)) \geqslant 1$ are obviously satisfied for non-narrow cuts C of x^*, and we saw that they do not appear in the description of $P^{\uparrow}_{Q_T\text{-join}}$ for narrow cuts C if $|T \cap \delta(C)|$ is odd. For the remaining narrow cuts C, property (i) implies $y(\delta(C)) \geqslant 1/\varepsilon$, and hence $z(\delta(C)) \geqslant \frac{1}{2}(1 + \varepsilon \cdot 1/\varepsilon) = 1$. $\qquad\square$

The close relation of y and T that is required in Observation 25 motivates studying τ-odd solutions, which to some extent embrace properties of y and T.

Definition 26 (τ-odd). *For $\tau \in \mathbb{Z}_{\geqslant 0}$ such that τ is odd, and a family $\mathcal{F}$ of cuts, we say that a point $y \in \mathbb{R}^E$ is τ-odd (with respect to $\mathcal{F}$), if for each $C \in \mathcal{F}$, either*

    (i) $y(\delta(C)) \leqslant \tau$, y is integral on the edges in $\delta(C)$, and $y(\delta(C))$ is odd, or

    (ii) $y(\delta(C)) \geqslant \tau + 2$.

We call the cuts C satisfying (i) and (ii), respectively, the y-small and y-large cuts.

Note that, for $\tau \approx 1/\varepsilon$, given a short τ-odd point $y \in P_{\mathrm{ST}}$ with respect to the narrow cuts $\mathcal{N}$ of an optimal solution x^* of $(\mathrm{LP_{HK}})$, a tree with the properties needed in Observation 25 could be obtained as a minimum length spanning tree T such that χ^T coincides with y on the integral edges of y. This reduces the problem to finding short τ-odd points, which is where our dynamic programming approach can help.

6.2 Obtaining τ-odd points via our DP

We now discuss how our dynamic programming approach can be adjusted to compute τ-odd points. Note that τ-odd points are by definition very similar to τ-integral points: The extra constraint is essentially that a τ-odd point can only have an odd number of edges in small cuts. Our dynamic programming approach can

---

[12] NP-hardness can for example be derived by a reduction from the Hamiltonian s-t path problem. For this consider an arbitrary numbering of the vertices $V = \{v_1, \ldots, v_n\}$ with $v_1 = s$ and $v_n = t$, and consider the complete chain $S_1, \ldots, S_{n-1}$, where $S_i = \{v_1, \ldots, v_i\}$ for $i \in [n-1]$. First, the cuts S_1 and S_{n-1} ensure that a spanning tree that is odd in each cut must have odd degree at v_1 and v_n. Moreover, for any $i \in \{2, \ldots, n-1\}$, the vertex v_i must have even degree in the spanning tree due to the cut constraints on S_{i-1} and S_i. Because the degrees of a spanning tree on n vertices sum up to $2(n-1)$, this implies that v_1 and v_n must have degree 1, and all other vertices degree 2. However, a spanning tree with these properties is a Hamiltonian v_1-v_n path, and any Hamiltonian v_1-v_n path is such a spanning tree.

easily handle this type of constraints, as edges in small cuts are always determined by the connectivity triples used. Thus, if we define the set

$$\mathcal{K}' := \left\{(S, F, \mathcal{C}) \in \mathcal{K} \,\middle|\, |F| \text{ odd}\right\} \cup \left\{(\emptyset, \emptyset, \{\emptyset\}), (V, \emptyset, \{\emptyset\})\right\}$$

and run the dynamic program presented in Section 3 with $\mathcal{K}'$ instead of $\mathcal{K}$ (and no lower or upper bounds on the cuts), we immediately obtain the following analogue of Theorem 7.

Theorem 27. *Let $\mathcal{C}$ be a chain of cuts. For any $\tau \in \mathbb{Z}_{\geqslant 0}$, there is an algorithm that returns in $|V|^{\mathcal{O}(\tau)}$ time a τ-odd point $y \in P_{\mathrm{ST}}$ with respect to $\mathcal{C}$ such that $c^{\top}y \leqslant c(T)$ for every spanning tree T that has an odd number of edges in every cut in $\mathcal{C}$.*

The above is precisely what we need for Path TSP: In this special case, we have $Q = \{s, t\}$, hence the family $\mathcal{N}$ of narrow cuts has width 1, i.e., it is a chain. This was also observed earlier by An, Kleinberg and Shmoys [1] and has been crucial in many recent improvements for Path TSP. In our case, it guarantees applicability of Theorem 27, and allows the approximation algorithm for Path TSP stated as Algorithm 5, which we analyze to prove Theorem 3 below.

Algorithm 5: Polynomial $(3/2 + \varepsilon)$-approximation for s-t Path TSP

1. Let x^* be an optimal solution of ($\mathrm{LP_{HK}}$), and let $\mathcal{C}$ be the family of narrow cuts of x^* not containing t.
2. Let $\tau \in \{\lfloor 1/\varepsilon \rfloor, \lfloor 1/\varepsilon \rfloor + 1\}$ odd, and use the algorithm guaranteed by Theorem 27 to find a τ-odd point $y \in P_{\mathrm{ST}}$ with respect to $\mathcal{C}$.
3. Let T be the shortest spanning tree of G such that χ^T coincides with y on all integral edges of y.
4. Let J be a shortest Q_T-join in G, and return the shortcutted multiunion of J and T.

Proof of Theorem 3. We claim that the pair (y, T) generated in Algorithm 5 has the properties listed in Observation 25. To see that point (i) holds, first note that by τ-oddness of y with respect to $\mathcal{C}$, we have that for every narrow cut C of x^*, either $y(\delta(C)) \geqslant \tau + 2 \geqslant 1/\varepsilon$, or y is integral on $\delta(C)$ and $y(\delta(C))$ is odd. As χ^T coincides with y on all integral edges of y, the latter case in particular implies that χ^T coincides with y on $\delta(C)$, and consequently, $|T \cap \delta(C)|$ is odd. Hence, point (i) is satisfied. For point (ii), observe that by definition, $c(T) \leqslant c^{\top}y$, and Theorem 27 implies $c^{\top}y \leqslant c(\mathrm{OPT})$. Together, this yields $c(T) \leqslant c(\mathrm{OPT})$, as desired.

By Observation 25, it follows that Algorithm 5 is a $(1.5 + \varepsilon)$-approximation for Path TSP. For a running time guarantee, we observe that all steps in Algorithm 5 can be performed efficiently. For step 1, note that the minimum cut in G with respect to weights x^* has value at least 1 by the lower bounds in the relaxation ($\mathrm{LP_{HK}}$), and enumerating all cuts with values that are within a factor of 2 from the minimum cut can be done in time $\mathcal{O}(|V|^4 |E|)$ (see [22]). Step 2 can be done in time $|V|^{\mathcal{O}(1/\varepsilon)}$ by Theorem 27. Finally, minimum length Q-joins can be obtained efficiently (see [12], for example), with running times dominated by the second step. Thus, the running time of Algorithm 5 is $|V|^{\mathcal{O}(1/\varepsilon)}$. This completes the proof of Theorem 4. $\qquad \square$

The above approach exploits that the τ-odd point y lies in the spanning tree polytope: This guarantees that a spanning tree T coinciding with y on integral edges can be found in step 3 of Algorithm 5. In the more general case of connected Q-joins with $|Q| > 2$, the narrow cuts $\mathcal{N}$ no longer form a chain, which causes problems in directly extending the propagation step of our dynamic programming approach, as we highlighted in the generalization from MCCST to MLCST. With minor modifications, ideas of this generalization also help for the connected Q-join problem. In fact, a simple alteration of the extension step can be used to obtain a pair (y, T) with the properties highlighted in the following theorem.

Theorem 28. *For any odd $\tau \in \mathbb{Z}_{\geqslant 0}$ and a laminar family $\mathcal{L}$ of width k, there is an algorithm that returns in time $|V|^{\mathcal{O}(k\tau)}$ a point $y \in \mathbb{R}_{\geqslant 0}^E$ and a spanning tree T of G with the following properties:*

   (i) y is τ-odd with respect to $\mathcal{L}$.
   (ii) χ^T coincides with y on all y-small cuts in $\mathcal{L}$.
   (iii) $\ell(T) \leqslant \ell^\top y \leqslant \ell^\top x$ for every τ-odd point $x \in P_{\mathrm{ST}} \cap \mathbb{Z}_{\geqslant 0}^E$.

Observe that Theorem 28 allows for bounding the length of the τ-good points that we find from above by the length of τ-good spanning trees only, but by Theorem 24, this is sufficient for comparing to optimal solutions of the Q-join problem, once we prove that connected Q-joins that are spanning trees are indeed τ-good. From the above ingredients, we obtain our approximation algorithm for the connected Q-join problem, which is stated as Algorithm 6 below. It is this algorithm that we analyze for a proof of Theorem 4.

Algorithm 6: Polynomial $(3/2 + \varepsilon)$-approximation for the shortest connected Q-join problem

---

1. Let x^* be an optimal solution of (LP$_{\mathrm{HK}}$), and let $\mathcal{N}$ be the family of all narrow cuts of x^* not containing a fixed element $q \in Q$.

2. Let $\tau \in \{\lfloor 1/\varepsilon \rfloor, \lfloor 1/\varepsilon \rfloor + 1\}$ be odd, and use the algorithm guaranteed by Theorem 28 to find a pair (y, T) of a τ-odd point $y \in \mathbb{R}_{\geqslant 0}^E$ and a spanning tree T of G.

3. Let J be a shortest Q_T-join in G, and return the multiunion of J and T.

Proof of Theorem 4. As in the proof of Theorem 3, it is easy to see that the pair (y, T) returned by Algorithm 6 satisfies point (i) in Observation 25. To prove that point (ii) holds, we claim that for any spanning tree S that is a Q-join, χ^S is a τ-good solution of P_{ST} with respect to $\mathcal{N}$. By Theorem 28, this implies that $\ell^\top y \leqslant \ell(S)$ for all spanning trees S that are Q-joins. By Theorem 24, at least one such spanning tree is in fact an optimal solution to the Q-join problem, and thus $\ell^\top y \leqslant \ell(\mathrm{OPT})$ follows. Furthermore, Theorem 28 also implies $\ell(T) \leqslant \ell^\top y \leqslant \ell(\mathrm{OPT})$.

For concluding the approximation guarantee of $3/2 + \varepsilon$ with the help of Observation 25, it is thus sufficient to prove the claim. Thereto, consider a spanning tree S that is a Q-join, and let $C \in \mathcal{N}$. We show that $\chi^S(\delta(C))$ is odd, which implies the claim due to integrality of χ^S. Double counting the edges in $S \cap E[C]$, we get

$$\chi^S(\delta(C)) = |S \cap \delta(C)| = \sum_{v \in C} \deg_S(v) - 2 \cdot |S \cap E[C]| \ .$$

Consequently, $\chi^S(\delta(C))$ has the same parity as the number of odd-degree vertices in C with respect to S. But as S is a Q-join, the latter number is equal to $|C \cap Q|$, and as we already observed previously, all $C \in \mathcal{N}$ satisfy that $|C \cap Q|$ is odd. This finishes the proof of the claim.

Finally, in order to obtain a running time bound, note that step 1 can be done in time $\mathcal{O}(|V|^4 |E|)$ (see the proof of Theorem 3). The second step has running time $|V|^{\mathcal{O}(k/\varepsilon)}$ by Theorem 28, and minimum length Q-joins can be obtained efficiently (see [12], for example), with running times dominated by the second step. Thus, the running time of Algorithm 6 is bounded by $|V|^{\mathcal{O}(|Q|/\varepsilon)}$. This completes the proof of Theorem 4. $\qquad \square$

It thus remains to give a proof of Theorem 28, which we outline in the remainder of this section. We adopt the dynamic programming approach used for MLCST, where in order to make sure that the DP guesses an odd number of edges in small cuts, we use $\mathcal{K}'$ instead of $\mathcal{K}$ throughout the procedure. For every connectivity triple $(S, F, \mathcal{C}) \in \mathcal{K}'$, the dynamic programming approach will construct a pair $(y_{(S,F,\mathcal{C})}, T_{(S,F,\mathcal{C})}) \in \mathbb{R}_{\geqslant 0}^E \times 2^E$ with the following property.

Property 29.

   (i) $\mathrm{supp}(y_{(S,F,\mathcal{C})}) \subseteq E[S]$, and $y_{(S,F,\mathcal{C})}$ is τ-odd with respect to $\mathcal{L}_S$.

29

(ii) $\chi^{T_{(S,F,\mathcal{C})}} \in P_{\mathrm{ST}}(S,F,\mathcal{C})$, *and* $\chi^{T_{(S,F,\mathcal{C})}}$ *coincides with* $y_{(S,F,\mathcal{C})}$ *on all* $y_{(S,F,\mathcal{C})}$-*small cuts in* $\mathcal{L}_S$.

(iii) *For all* $U \subseteq E[S]$ *such that* $\chi^U \in P_{\mathrm{ST}}(S,F,\mathcal{C})$ *and* $\chi^U + \chi^F$ *is* τ-*odd with respect to* $\mathcal{L}_S$, *we have*
$$\ell(T_{(S,F,\mathcal{C})}) \leqslant \ell^\top y_{(S,F,\mathcal{C})} \leqslant \ell(U).$$

It is clear that if we can construct such pairs for all triples $(S,F,\mathcal{C}) \in \mathcal{K}'$, then we are done, as $(y_{(V,\emptyset,\{\emptyset\})}, T_{(V,\emptyset,\{\emptyset\})})$ satisfies the assumptions of Observation 25. To maintain pairs (y, T) in the dynamic program, we replace the extension step (Algorithm 3) by the one presented in Algorithm 7 below.

Algorithm 7: Extending $(y_{(S_1,F_1,\mathcal{C}_1)}, T_{(S_1,F_1,\mathcal{C}_1)}), \ldots, (y_{(S_w,F_w,\mathcal{C}_w)}, T_{(S_w,F_w,\mathcal{C}_w)})$ to $(S,F,\mathcal{C})$.

1. Let z be a minimizer of the linear program

$$
\begin{aligned}
\min \quad & \ell^\top z && \text{(lamExLP}_2\text{)}\\
& z \in P_{\mathrm{ST}}(S,F,\mathcal{C}) \\
z(\delta(S')) + |F \cap \delta(S')| \geqslant {}& \tau + 2 && \forall S \in \mathcal{L} \text{ such that } \exists i \in [w] \text{ with } S_i \subsetneq S' \subsetneq S \\
z(e) = {}& \chi^{T_{(S_i,F_i,\mathcal{C}_i)}}(e) && \forall e \in E[S_i], \ \forall i \in [w] \\
z(e) = {}& \chi^{F_i}(e) && \forall e \in \delta(S_i) \setminus F, \ \forall i \in [w] \ .
\end{aligned}
$$

2. Define $y \in \mathbb{R}_{\geqslant 0}^E$ by $y(e) = \begin{cases} y_{(S_i,F_i,\mathcal{C}_i)}(e) & \text{if } e \in E[S_i] \text{ for some } i \in [w], \\ z(e) & \text{else.} \end{cases}$

3. Let $T \subseteq E[S]$ be of minimum length $\ell(T)$ such that
   (i) $\chi^T \in P_{\mathrm{ST}}(S,F,\mathcal{C})$, (ii) $T \cap E[S_i] = T_{(S_i,F_i,\mathcal{C}_i)} \ \forall i \in [w]$, and (iii) $T \cap \delta(S_i) = F_i \ \forall i \in [w]$.

4. Output (y, T).

The idea of this modified extension step is to maintain both a fractional point and a spanning tree at every stage, where (as in the dynamic program for MLCST) extension steps using the linear program are always based on trees. A new fractional point is then obtained by combining the potentially fractional extension with the fractional solutions of the subproblems (see step 2). Opposed to the situation in MLCST, the application here only requires spanning trees that have an odd number of edges in small cuts of the corresponding fractional point. This is easily achieved by the construction of the trees in step 3. This construction guarantees that every pair (y, T) returned by Algorithm 7 satisfies the first two points of Property 29 with respect to the connectivity triple $(S,F,\mathcal{C})$, which we formally prove in Lemma 30.

For propagation, we use Algorithm 8, which is an analogon of Algorithm 4. It considers all potential candidates for $(y_{(S,F,\mathcal{C})}, T_{(S,F,\mathcal{C})})$ that can be obtained through extension steps, and the shortest pair (y, T) with respect to $\ell^\top y$ is returned.

Algorithm 8: Propagation to $(y_{(S,F,\mathcal{C})}, T_{(S,F,\mathcal{C})})$ from all $(y_{(S',F',\mathcal{C}')}, T_{(S',F',\mathcal{C}')})$ with $S' \subsetneq S$.

1. For every choice of triples $(S_i, F_i, \mathcal{C}_i) \in \mathcal{K}'$ for $i \in [w]$ where $S_1, \ldots, S_w \subsetneq S$ have pairwise empty intersections, apply Algorithm 7 to extend $(y_{(S_1,F_1,\mathcal{C}_1)}, T_{(S_1,F_1,\mathcal{C}_1)}), \ldots, (y_{(S_w,F_w,\mathcal{C}_w)}, T_{(S_w,F_w,\mathcal{C}_w)})$ to $(S,F,\mathcal{C})$. Collect all pairs (y, T) obtained this way in the set $\mathcal{P}$.

2. Return $(y_{(S,F,\mathcal{C})}, T_{(S,F,\mathcal{C})}) \in \mathrm{argmin}_{(y,T) \in \mathcal{P}} \ell^\top y$.

We now show that the pairs (y, T) returned by Algorithm 8 have Property 29. Adapting the approach pursued in the case of MLCST (Lemma 22), we proceed by induction. Obviously, the pair $(y_{(\emptyset,\emptyset,\{\emptyset\})}, T_{(\emptyset,\emptyset,\{\emptyset\})}) = (0, \emptyset)$ satisfies Property 29, and the inductive step is given by the following lemma.

Lemma 30. *Let $(S, F, C) \in \mathcal{K}'$ with $S \neq \emptyset$. Assume that for all $(S', F', C') \in \mathcal{K}'$ with $S' \subsetneq S$, we are given $(y_{(S', F', C')}, T_{(S', F', C')})$ satisfying Property 29, and let $(y_{(S, F, C)}, T_{(S, F, C)})$ be obtained from Algorithm 8. Then $(y_{(S, F, C)}, T_{(S, F, C)})$ satisfies Property 29, as well.*

Proof. Let $\mathcal{P}$ be defined as in Algorithm 8. We already remarked above that all pairs $(y, T) \in \mathcal{P}$ satisfy point (i) and point (ii) of Property 29 with respect to the connectivity triple (S, F, C). Indeed, $\mathrm{supp}(y_{(S, F, C)}) \subseteq E[S]$ holds by definition, as well as $\chi^{T_{(S,F,C)}} \in P_{\mathrm{ST}}(S, F, C)$. Additionally, the facts that $y_{(S, F, C)}$ is τ-odd with respect to $\mathcal{L}_S$ and that $\chi^{T_{(S,F,C)}}$ coincides with $y_{(S, F, C)}$ on all $y_{(S, F, C)}$-small cuts in $\mathcal{L}_S$ are true by definition for cuts $L \in \mathcal{L}_S$ with $S_i \subsetneq L$ for some $i \in [w]$; and they are implied by the assumptions on $(y_{(S_i, F_i, C_i)}, T_{(S_i, F_i, C_i)})$ through Property 29 for cuts $L \in \mathcal{L}_S$ with $L \subsetneq S_i$ for some $i \in [w]$.

Moreover, for any pair $(y, T) \in \mathcal{P}$ we also have $\ell(T) \leqslant \ell^\top y$: If z is the solution of the linear program (lamExLP$_2$) that was used to define y, then $\ell(T) \leqslant \ell^\top z$ because χ^T is in fact an optimal solution of the same linear program without the constraints $z(\delta(S)) \geqslant \tau + 2$. As by assumption, $\ell(T_{(S_i, F_i, C_i)}) \leqslant \ell^\top y_{(S_i, F_i, C_i)}$, we further see that $\ell^\top z \leqslant \ell^\top y$, and hence $\ell(T) \leqslant \ell^\top y$, which is the first statement in point (iii).

Thus, it remains to prove that there exists a pair $(y, T) \in \mathcal{P}$ such that $\ell^\top y \leqslant \ell(U)$ for every set $U \subseteq E[S]$ such that $\chi^U \in P_{\mathrm{ST}}(S, F, C)$ and $\chi^U + \chi^F$ is τ-odd with respect to $\mathcal{L}_S$. Fix such a set U, and let $R \subseteq \binom{V \setminus S}{2}$ be any set of edges that is right-compatible with (S, F, C). Then, $T := U \cup F \cup R$ is in the spanning tree polytope of $(V, E \cup R)$. Let $S_1, \ldots, S_w \subsetneq S$ be the maximal χ^T-small cuts in $\mathcal{L}_S$, and let $F_i = T \cap \delta(S_i)$. We define connectivity patterns $C_1, \ldots, C_w$ such that $(S_i, F_i, C_i) \in \mathcal{K}'$ iteratively as follows, where $T_0 := T$ and $i \in [w]$.

$$\text{Let } C_i \text{ s.t. } T_{i-1} \text{ is compatible with } (S_i, F_i, C_i), \text{ and let } T_i := \left(T_{i-1} \setminus E[S_i]\right) \cup T_{(S_i, F_i, C_i)}. \tag{12}$$

We claim that if extending $(y_{(S_1, F_1, C_1)}, T_{(S_1, F_1, C_1)}), \ldots, (y_{(S_w, F_w, C_w)}, T_{(S_w, F_w, C_w)})$ to (S, F, C) using Algorithm 7 returns the pair (y, T), then $\ell^\top y \leqslant \ell(U)$. To this end, observe that if z is the solution of the linear program (lamExLP$_2$) used in this call to Algorithm 7, then we can write

$$\ell^\top y = \sum_{i \in [w]} \ell^\top y_{(S_i, F_i, C_i)} + \ell^\top z - \sum_{i \in [w]} \ell(T_{(S_i, F_i, C_i)}) \ . \tag{13}$$

We will bound the right-hand side by $\ell(U)$. Thereto, we claim that by the construction in (12), we have $\ell^\top y_{(S_i, F_i, C_i)} \leqslant \ell(U \cap E[S_i])$. This follows from invoking Property 29 (iii) for the pair $(y_{(S_i, F_i, C_i)}, T_{(S_i, F_i, C_i)})$ (which is satisfied by assumption) with the edge set $U \cap E[S_i]$. To this end, we have to show that (a) $\chi^{U \cap E[S_i]} \in P_{\mathrm{ST}}(S_i, F_i, C_i)$, and (b) $\chi^{U \cap E[S_i]} + \chi^{F_i}$ is τ-odd with respect to $\mathcal{L}_{S_i}$. Indeed, (a) follows from the fact that T_{i-1} is compatible with (S_i, F_i, C_i), hence $\chi^{T_{i-1} \cap E[S_i]} \in P_{\mathrm{ST}}(S_i, F_i, C_i)$, and $T_{i-1} \cap E[S_i] = U \cap E[S_i]$ by construction; (b) follows from $\chi^U + \chi^F$ being τ-odd with respect to $\mathcal{L}_S$, as $S_i \subsetneq S$. Furthermore, note that T_w is compatible with (S, F, C), hence $T_w \cap E[S]$ is left-compatible with (S, F, C), i.e., $\chi^{T_w \cap E[S]} \in P_{\mathrm{ST}}(S, F, C)$. Moreover, we can write

$$T_w \cap E[S] = \left(U \setminus \bigcup_{i \in [w]} E[S_i]\right) \cup \bigcup_{i \in [w]} T_{(S_i, F_i, C_i)} \ ,$$

and from this form and the fact that $S_1, \ldots, S_w$ are maximal χ^T-small cuts in $\mathcal{L}_S$, it can be seen that $T_w \cap E[S]$ is an integral solution of (lamExLP$_2$). As z is an optimal fractional solution of the same linear program, we have $\ell^\top z \leqslant \ell(T_w \cap E[S])$. Applying the inequalities that we just obtained to (13), we get

$$\begin{aligned}
\ell^\top y &\leqslant \sum_{i \in [w]} \ell(U \cap E[S_i]) + \ell(T_w \cap E[S]) - \sum_{i \in [w]} \ell(T_{(S_i, F_i, C_i)}) \\
&= \sum_{i \in [w]} \ell(U \cap E[S_i]) + \ell\left(U \setminus \bigcup_{i \in [w]} E[S_i]\right) \\
&= \ell(U) \ ,
\end{aligned}$$

as desired. □

In order to bound the running time of our dynamic program, we need an upper bound on the number of connectivity triples in $\mathcal{K}'$, but this is easily obtained from the upper bound $|\mathcal{K}| \leqslant |V|^{\mathcal{O}(\tau)}$ in Proposition 11: We have $\mathcal{K}' \subseteq \mathcal{K}$, and thus also $|\mathcal{K}'| \leqslant |V|^{\mathcal{O}(\tau)}$. With these ingredients, we are finally ready to formally prove Theorem 28.

Proof of Theorem 28. We run a dynamic program that calculates pairs $(y_{(S,F,\mathcal{C})}, T_{(S,F,\mathcal{C})})$ for all $(S, F, \mathcal{C}) \in \mathcal{K}'$ starting with the initialization $(y_{(\emptyset,\emptyset,\{\emptyset\})}, T_{(\emptyset,\emptyset,\{\emptyset\})}) = (0, \emptyset)$, and using Algorithm 8 for propagation in an order such that $(y_{(S',F',\mathcal{C}')}, T_{(S',F',\mathcal{C}')})$ is computed before $(y_{(S,F,\mathcal{C})}, T_{(S,F,\mathcal{C})})$ if $S' \subsetneq S$. Note that $(y_{(\emptyset,\emptyset,\{\emptyset\})}, T_{(\emptyset,\emptyset,\{\emptyset\})})$ satisfies Property 29, and hence by an inductive application of Lemma 30, all pairs $(y_{(S,F,\mathcal{C})}, T_{(S,F,\mathcal{C})})$ satisfy Property 29. In particular, $(y_{(V,\emptyset,\{\emptyset\})}, T_{(V,\emptyset,\{\emptyset\})})$ is thus a pair satisfying the guarantees of Theorem 28.

In terms of running time, the dominating operation is repeatedly solving linear programs of the type (lamExLP$_2$). The total number of linear programs that we have to solve during this procedure is bounded from above by $|\mathcal{K}'|^{k+1}$, and the running time of $|V|^{\mathcal{O}(k\tau)}$ thus follows from $\mathcal{K}' \subseteq \mathcal{K}$ and Proposition 11, as remarked above, and the fact that linear programs of the type (lamExLP$_2$) can be solved in strongly polynomial time by using a compact extended formulation for the spanning tree polytope with small coefficients in the constraint matrix (one can, for example, use the one by Martin [21], which has coefficients that are bounded by 1 in absolute value), and then applying the framework of Tardos [29]. □

A Weakness of the natural relaxation

In this section, we demonstrate that the natural relaxation of MCCST, which is given by

$$Q = \{x \in P_{\mathrm{ST}} \mid a_i \leqslant x(\delta(S_i)) \leqslant b_i \ \forall i \in [k]\} \ ,$$

is too weak for allowing small bounds on constraint violation when comparing an integral solution to fractional solutions of the relaxation. More precisely, we show the following.

Theorem 31. *For every $\varepsilon > 0$, there is an instance of the MCCST problem such that there exists a point $y \in Q$, but for any tree T satisfying the chain constraints, there is a cut C in the chain such that*

$$|T \cap \delta(C)| \geqslant (2 - \varepsilon) \cdot y(\delta(C)) \ .$$

Proof. We construct a family of instances of the MCCST problem depending on a parameter $k \in \mathbb{Z}_{>0}$, where each instance has the properties listed in Theorem 31, but with a factor $2 - 2/(k+2)$ instead of $2 - \varepsilon$. This is obviously enough to conclude the theorem because for a fixed $\varepsilon > 0$ and large enough k, we obtain an instance with the desired properties.

For $k \in \mathbb{Z}_{>0}$, let H_k be the graph obtained as follows. Start with a path of length 2^{k-1} on vertices $v_0, v_1, \ldots, v_{2^{k-1}}$, and for all $i, j \in \{0, 1, \ldots, 2^{k-1}\}$ such that $j - i = 2^\ell$ for some $\ell \in \mathbb{Z}_{\geqslant 0}$, add a path of length 2 between v_i and v_j. More precisely, for any such i and j, we add a new vertex $w_{i,j}$ as well as edges $\{v_i, w_{ij}\}$ and $\{w_{ij}, v_j\}$. Additionally, we define cuts S_ℓ for $\ell \in [2^k]$ by

$$S_\ell := \{v_i \mid 2i < \ell\} \cup \{w_{i,j} \mid i + j < \ell\} \ .$$

Note that for $\ell_1 < \ell_2$, we have $S_{\ell_1} \subseteq S_{\ell_2}$, and thus the family $\mathcal{S}_k := \{S_1, S_2, \ldots, S_{2^k}\}$ is a chain. An illustration of this construction for $k = 3$ is given in Figure 5.
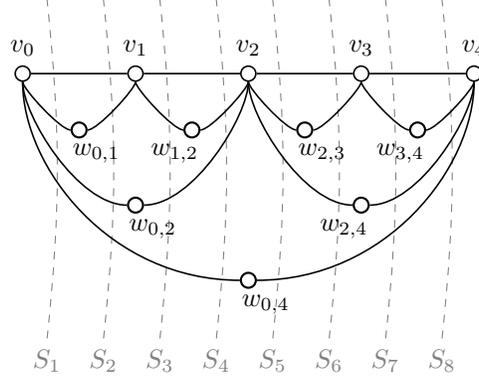
Figure 5: The graph H_k and the family $\mathcal{S}_k = \{S_1, \ldots, S_{2^k}\}$ of cuts for $k = 3$.

To complete the instances of the MCCST problem and to ensure feasibility, we can define $a_\ell = 0$ and $b_\ell = k + 1$ for all $\ell \in [2^k]$. Note that this is equivalent to not putting any constraints on the sizes of the given cuts. In fact our arguments are independent of the precise degree bounds (given feasibility).

We first observe that the corresponding relaxation Q has solutions with small weight on all cuts in $\mathcal{S}_k$. To this end, define $y \in \mathbb{R}_{\geqslant 0}^E$ by

$$y(e) = \begin{cases} 1 & \text{if } e = \{v_{i-1}, v_i\} \text{ for some } i \in [2^{k-1}], \\ 1/2 & \text{else.} \end{cases}$$

Note that y is indeed a point in P_{ST}. This can be seen by writing $y = {(x_1 + x_2)}/2$, where $x_1, x_2 \in \mathbb{R}_{\geqslant 0}^E$ are incidence vectors of spanning trees given by

$$x_1(e) = \begin{cases} 1 & \text{if } e = \{v_{i-1}, v_i\} \text{ for some } i, \\ 1 & \text{if } e = \{v_i, w_{i,j}\} \text{ for some } i, j, \\ 0 & \text{if } e = \{w_{i,j}, v_j\} \text{ for some } i, j, \end{cases} \quad \text{and} \quad x_2(e) = \begin{cases} 1 & \text{if } e = \{v_{i-1}, v_i\} \text{ for some } i, \\ 0 & \text{if } e = \{v_i, w_{i,j}\} \text{ for some } i, j, \\ 1 & \text{if } e = \{w_{i,j}, v_j\} \text{ for some } i, j. \end{cases}$$

Moreover, we observe that for every $i \in [2^k]$, we have

$$y(\delta(S_i)) = 1 + \frac{k}{2} \ . \tag{14}$$

The above implies that indeed, $y \in Q$. Now, consider a spanning tree T of H_k. We claim that for every $k \in \mathbb{Z}_{>0}$, there exists $i \in [2^k]$ such that

$$|T \cap \delta(S_i)| \geqslant k \ . \tag{15}$$

Once we prove this, we can combine (14) and (15) to obtain that there exists $i \in [2^k]$ such that

$$|T \cap \delta(S_i)| \geqslant \frac{k}{1 + k/2} \cdot y(\delta(S_i)) = \left(2 - \frac{2}{k+2} \right) \cdot y(\delta(S_i)) \ .$$

By choosing k large enough such that $\varepsilon \geqslant {2}/{(k+2)}$, we thus indeed obtain an instance satisfying the properties listed in Theorem 31. It remains to prove the claim. To this end, we show the following stronger lemma.

Lemma 32. *Let F be a subset of the edges of H_k such that any vertex $w_{i,j}$ of H_k is incident to at least one edge of F. Then there exists $i \in [2^k]$ such that $|F \cap \delta(S_i)| \geqslant k$.*

Proof of Lemma 32. We proceed by induction on k. The statement of the base case $k = 1$ is directly implied by the assumption on F. Indeed, at least one of the two edges $\{v_0, w_{0,1}\}$ and $\{w_{0,1}, v_1\}$ is in F, and correspondingly, at least one of $|F \cap \delta(S_1)| \geqslant 1$ or $|F \cap \delta(S_2)| \geqslant 1$ holds.

For the inductive step, let $k \geqslant 2$ and consider the graph $H_k = (V, E)$. By the assumption on F, at least one of the edges $\{v_0, w_{0,2^{k-1}}\}$ and $\{w_{0,2^{k-1}}, v_{2^{k-1}}\}$ is in F. By symmetry, we can assume without loss of generality that $\{v_0, w_{0,2^{k-1}}\} \in F$. Observe that the subgraph of H_k induced by the vertex set $V_0 := \{v_i \mid i \leqslant 2^{k-2}\} \cup \{w_{i,j} \mid i, j \leqslant 2^{k-2}\}$ is isomorphic to H_{k-1}, and $F \cap E[V_0]$ has an edge incident to every vertex $w_{i,j}$ of this copy of H_{k-1}. Hence, by induction, there exists $i \in [2^{k-1}]$ such that $|(F \cap E[V_0]) \cap \delta(S_i)| \geqslant k - 1$. As additionally, $\{v_0, w_{0,2^{k-1}}\} \in \delta(S_i)$, this implies $|F \cap \delta(S_i)| \geqslant k$. $\quad\square$

Finally, observe that by connectivity, every spanning tree T of H_k contains at least one edge incident to $w_{i,j}$, for all vertices $w_{i,j}$ of H_k. Consequently, Lemma 32 does indeed imply existence of $i \in [2^k]$ such that $|T \cap \delta(S_i)| \geqslant k$. This completes the proof of Theorem 31. $\quad\square$

B Analyzing the DP by backtracing OPT fails in the general case

The aim of this section is to extend an example from Section 5 which showed that the analysis of our dynamic programming approach cannot be done in the classical way, i.e., by backtracing an optimal solution. While the issues in the example constructed in Section 5 can be fixed by breaking ties in the right way, we now present a slightly more involved instance (see Figure 6) where the naive approach faces problems that cannot be avoided easily.
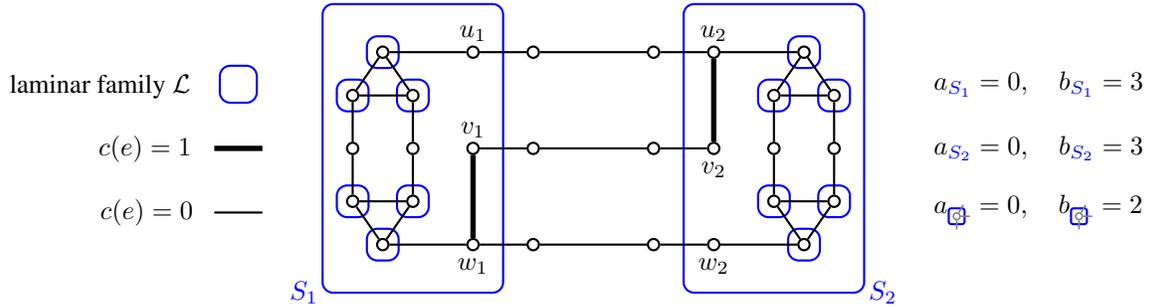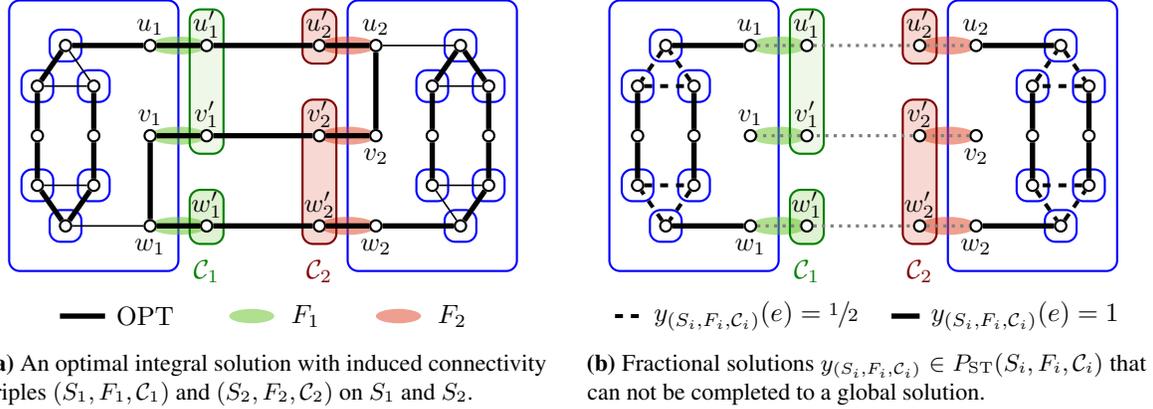

Figure 6: An instance of the MLCST problem: The laminar family $\mathcal{L}$ is given by the blue sets, with lower and upper bounds indicated on the right. Edge costs $c\colon E \to \mathbb{R}_{\geqslant 0}$ are 0 on all edges except for $c((v_1, w_1)) = c((u_2, v_2)) = 1$.

The problem instance in Figure 6 is very similar to the instance discussed in Section 5 (Figure 3). While the latter had edges (u_i, w_i) for $i \in \{1, 2\}$, the vertices u_i and w_i are connected by an auxiliary graph in the new instance. This auxiliary graph has the following two crucial properties:

(i) The auxiliary graph does not contain a spanning tree that satisfies the laminar constraints.

(ii) The spanning tree polytope of the auxiliary graph contains a point that satisfies the laminar constraints.

In other words, the two properties state that it is possible to "fractionally connect" u_i and w_i in the auxiliary graph, while an integral solution cannot connect u_i and w_i through the auxiliary graph. In particular, this implies that any feasible integral solution will use both edges (v_1, w_1) and (u_2, v_2), and thus have cost at least 2. One such integral solution is given in Figure 7a. Observe that no matter how we choose an integral solution, the connectivity patterns induced on the sets S_1 and S_2 will always be $(S_1, F_1, \mathcal{C}_1)$ and $(S_2, F_2, \mathcal{C}_2)$, respectively, as indicated in Figure 7a.

To analyze our DP approach by classical backtracing of an optimal solution, our goal would be to find partial solutions $y_{(S_i, F_i, \mathcal{C}_i)} \in P_{\mathrm{ST}}(S_i, F_i, \mathcal{C}_i)$ for $i \in \{1, 2\}$, and show that a common extension of these

(a) An optimal integral solution with induced connectivity triples (S_1, F_1, C_1) and (S_2, F_2, C_2) on S_1 and S_2.

(b) Fractional solutions $y_{(S_i, F_i, C_i)} \in P_{\mathrm{ST}}(S_i, F_i, C_i)$ that can not be completed to a global solution.

Figure 7: Optimal integral and partial fractional solutions connect vertices differently inside S_1 and S_2.

solutions has smaller value than the actual optimal solution we started with. In our example, however, property (ii) of the auxiliary graph allows for partial fractional solutions $y_{(S_i, F_i, C_i)}$ that differ substantially from integral solutions in terms of connectivity. More precisely, the two fractional solutions $y_{(S_1, F_1, C_1)}$ and $y_{(S_2, F_2, C_2)}$ given in Figure 7b are both of cost 0 (and hence optimal), but there does not exist a common extension that is feasible for the natural linear relaxation of our problem instance at all.

There is one last caveat that has to be addressed: Our dynamic program is designed to construct partial solutions inside all *small* cuts of the laminar family for any choice of edges in the small cuts and corresponding connectivity patterns, and it always extends previously found solutions. Above, the threshold τ for deciding whether a cut is small or large was implicitly assumed to be at least 3 so that both S_1 and S_2 are small cuts. In the particular example, this implies that all the other cuts in $\mathcal{L}$ (which are precisely the singleton cuts) would be small cuts, as well, forcing our dynamic program to first construct partial solutions in these small cuts and only then extend to S_1 and S_2. This would inevitably lead to integral partial solutions only, hence we do need a setting where the singleton cuts are *large* cuts.

To achieve this, we introduce dummy edges that increase the number of edges in the singleton cuts of $\mathcal{L}$. More precisely, for any given threshold τ and every small singleton cut $\{x\} \in \mathcal{L}$, we can modify the problem instance as follows to turn $\{x\}$ into a large cut: Introduce new vertices $x_1, \ldots, x_\tau$ and edges $\{x, x_i\}$ for $1 \leqslant i \leqslant \tau$, and increase the bounds $a_{\{x\}}$ and $b_{\{x\}}$ by τ. Feasible solutions of the old and the new instance are in one-to-one correspondence and can be transformed into one another by adding or removing all the edges $\{x, x_i\}$, which are obviously part of any feasible solution of the new instance.

To conclude, by introducing dummy edges in the graph given in Figure 6 as described above, we obtain an instance where an analysis of our DP approach by backtracing an optimal solution fails inevitably, thus again emphasizing the importance of our novel approach.

References

[1] H.-C. An, R. Kleinberg, and D. B. Shmoys. Improving Christofides' algorithm for the s-t path TSP. *Journal of the ACM*, 62(5):34:1–34:28, 2015.

[2] A. Asadpour, M. X. Goemans, A. Madry, S. Oveis Gharan, and A. Saberi. An $O(\log n / \log \log n)$-approximation algorithm for the Asymmetric Traveling Salesman Problem. In *Proceedings of the 21st ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 379–389, 2010.

[3] N. Bansal, R. Khandekar, J. Könemann, V. Nagarajan, and B. Peis. On generalizations of network design problems with degree bounds. *Mathematical Programming, Series A*, 141:479–506, 2013.

[4] N. Bansal, R. Khandekar, and V. Nagarajan. Additive guarantees for degree-bounded directed network design. *SIAM Journal on Computing*, 39(4):1413–1431, 2009.

[5] K. Chaudhuri, S. Rao, S. Riesenfeld, and K. Talwar. A push-relabel approximation algorithm for approximating the minimum-degree MST problem and its generalization to matroids. *Theoretical Computer Science*, 410:4489–4503, 2009.

[6] K. Chaudhuri, S. Rao, S. Riesenfeld, and K. Talwar. What would Edmonds do? Augmenting paths and witnesses for degree-bounded MSTs. *Algorithmica*, 55:157–189, 2009.

[7] C. Chekuri, J. Vondrák, and R. Zenklusen. Dependent randomized rounding via exchange properties of combinatorial structures. *CoRR*, abs/0909.4348, 2009.

[8] C. Chekuri, J. Vondrák, and R. Zenklusen. Dependent randomized rounding via exchange properties of combinatorial structures. In *51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 575–584, 2010.

[9] J. Cheriyan, Z. Friggstad, and Z. Gao. Approximating minimum-cost connected T-joins. *Algorithmica*, 72:126–147, 2015.

[10] N. Christofides. Worst-case analysis of a new heuristic for the Travelling Salesman Problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie Mellon University, 1976.

[11] J. Edmonds. Matroids and the greedy algorithm. *Mathematical Programming*, 1(1):127–136, 1971.

[12] J. Edmonds and E. L. Johnson. Matching, Euler tours and the chinese postman. *Mathematical Programming*, 5(1):88–124, 1973.

[13] L. A. Goddyn. Some open problems I like. http://people.math.sfu.ca/~goddyn/Problems/problems.html, 2004.

[14] M. X. Goemans. Minimum bounded degree spanning trees. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 273–282, 2006.

[15] C. Gottschalk and J. Vygen. Better s-t-tours by Gao trees. In *Proceedings of 18th International Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 126–137, 2016.

[16] K. Kawarabayashi, Y. Kobayashi, and B. Reed. The disjoint paths problem in quadratic time. *Journal of Combinatorial Theory, Series B*, 102(2):424 – 435, 2012.

[17] J. Könemann and R. Ravi. A matter of degree: Improved approximation algorithms for degree-bounded minimum spanning trees. In *32nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 537–546, 2000.

[18] J. Könemann and R. Ravi. Primal-dual meets local search: Approximating MSTs with nonuniform degree bounds. In *35th Annual ACM Symposium on Theory of Computing (STOC)*, pages 389–395, 2003.

[19] B. Korte and J. Vygen. *Combinatorial Optimization, Theory and Algorithms*. Springer, 6th edition, 2018.

[20] A. Linhares and C. Swamy. Approximating min-cost chain-constrained spanning trees: A reduction from weighted to unweighted problems. In *Proceedings of the 18th Conference on Integer Programming and Combinatorial Optimization (IPCO)*, 2016.

[21] R. Martin. Using separation algorithms to generate mixed integer model reformulations. *Operations Research Letters*, 10:119–128, 1991.

[22] H. Nagamochi, K. Nishimura, and T. Ibaraki. Computing all small cuts in an undirected network. *SIAM Journal in Discrete Mathematics*, 10(3):469–481, 1997.

[23] N. Olver and R. Zenklusen. Chain-constrained spanning trees. *Mathematical Programming, Series A*, 167(2):293–314, 2018.

[24] A. Schrijver. *Combinatorial Optimization – Polyhedra and Efficiency*. Springer, 2003.

[25] A. Sebő. Eight-fifth approximation for the path TSP. In *Proceedings of 16th International Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 263–373, 2013.

[26] A. Sebő and A. van Zuylen. The salesman's improved paths: A $3/2 + 1/34$ approximation. In *Proceedings of 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 118–127, 2016.

[27] M. Singh and L. C. Lau. Approximating minimum bounded degree spanning trees to within one of optimal. In *39th Annual ACM Symposium on Theory of Computing (STOC)*, pages 661–670. ACM, 2007.

[28] O. Svensson, J. Tarnawski, and L. A. Végh. A constant-factor approximation algorithm for the Asymmetric Traveling Salesman Problem. In *Proceedings of 50th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 204–213, 2018.

[29] E. Tardos. A strongly polynomial algorithm to solve combinatorial linear programs. *Operations Research*, 34(2):250–256, 1986.

[30] V. Traub and J. Vygen. Approaching $\frac{3}{2}$ for the s-t path TSP. In *Proceedings of the 29th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1854–1864, 2018.

[31] J. Vygen. Reassembling trees for the traveling salesman. *SIAM Journal on Discrete Mathematics*, 30(2):875–894, 2016.

[32] L. A. Wolsey. Heuristic analysis, linear programming and branch and bound. *Mathematical Programming Studies*, 13:121–134, 1980.

[33] R. Zenklusen. Matroidal degree-bounded minimum spanning trees. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1512–1521, 2012.

[34] R. Zenklusen. A 1.5-approximation for Path TSP. In *Proceedings of the 30th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1539–1549, 2019.